Memory analysis of the KBeast Linux rootkit

Investigating publicly available Linux rootkit using the Volatility memory analysis framework

Richard Carbone
DRDC – Valcartier Research Centre

Defence Research and Development Canada

Scientific Report
DRDC-RDDC-2015-R064
June 2015

Abstract

This report is the first in a series examining Linux Volatility-specific memory malware-based analysis techniques. With minimal use of scanner-based technologies, the author will demonstrate what to look for while conducting Linux-based memory investigations using Volatility. This investigation consists of a memory image infected by the KBeast rootkit that will be analysed using Volatility. Through the proper application of various Volatility plugins combined with an in-depth knowledge of the Linux operating system, this case study can provide guidance to other investigators in their own Linux-based memory analyses.

Significance to defence and security

Canadian Armed Forces' (CAF) networks are a choice target for malware and directed attacks. This series of reports will provide junior and senior incident handlers alike with the necessary knowledge to investigate and mitigate complex attacks using only a memory image and a functional knowledge of the Linux operating system. As Linux continues to play a more important role in IT and the data centres of the Government of Canada and National Defence, some of these systems will invariably become infected. Thus, when this happens and when analysts and incident handlers have to intervene, it is hoped that these reports will have helped them to prepare for just such an occasion.

This page intentionally left blank.

Résumé

Ce rapport est le premier d'une série examinant les techniques spécifiques d'analyse de logiciels malveillants en mémoire sous Linux à l'aide de l'outil Volatility. En utilisant au minimum les technologies basées sur des scans, l'auteur démontrera ce qu'il faut rechercher lors d'une investigation de la mémoire Linux avec Volatility. Cette investigation avec Volatility sera effectuée sur une image mémoire infectée par le programme malveillant furtif KBeast. Nous espérons qu'en combinant correctement les différents greffons de Volatility et avec une connaissance approfondie du système d'exploitation Linux, cette étude de cas servira de guide à d'autres investigateurs dans leur propre analyse de mémoire Linux.

Importance pour la défense et la sécurité

Les réseaux des Forces armées canadiennes (FAC) sont une cible de choix pour les logiciels malveillants et les attaques dirigées. Cette série de rapports fournira aux analystes en réponse aux incidents, aussi bien juniors que séniors, toute la connaissance requise pour investiguer et mitiger des attaques complexes en utilisant seulement une image de la mémoire et une connaissance fonctionnelle du système d'exploitation Linux. Comme Linux joue un rôle de plus en plus important dans les TI et les centres de données du gouvernement du Canada et de la Défense nationale, certains de ces systèmes deviendront invariablement infectés. Par conséquent, quand cela arrivera et que des analystes en réponses aux incidents auront à intervenir, nous espérons que ces rapports les auront aidés à se préparer à une telle occasion.

This page intentionally left blank.

Table of contents

List of tables

Acknowledgements

The author would like to thank Mr. Francois Rheaume, Defence Scientist, for conducting both a preliminary and peer review of this text as well as providing helpful comments in order to improve it. Moreover, the author would also like to extend his thanks to Mr. Martin Salois, Defence Scientist, for improvements to the beginning portions of this text and for translating portions of this text.

Disclaimer policy

The content of this report is not advice and should not be treated as such.

Her Majesty the Queen in right of Canada, as represented by the Minister of National Defence ("Canada"), makes no representations or warranties, express or implied, of any kind whatsoever, and assumes no liability for the accuracy, reliability, completeness, currency or usefulness of any information, product, process or material included in this report. Moreover, nothing in this report should be interpreted as an endorsement of the specific use of any of the tools or techniques examined in it.

Any reliance on, or use of, any information, product, process or material included in this report is at the sole risk of the person so using it or relying on it.

Canada does not assume any liability in respect of any damages or losses arising out of or in connection with the use of, or reliance on, any information, product, process or material included in this report.

Requirements, assumptions and exclusions

The author assumes that the reader is familiar with digital forensics and the various techniques and methodologies associated therein. This report is not an introduction to digital forensics or to said techniques and methodologies. However, the author has endeavoured to ensure that the reader would have the information necessary to carry out a forensic analysis of a computer memory image suspected of malware infection based on the information and techniques described herein.

The experimentation conducted throughout this report was carried out atop a Fedora 20 64-bit Linux operating system. Unlike the various Windows infected memory case studies, neither anti-virus (AV) nor data carving techniques worked particularly well against Linux-based memory images. As such, the former is used minimally while the latter is not at all used in this report. Consequently, the methodology presented in this series of reports is quite different from that presented in the Windows Volatility-based series of memory malware analyses.

It is important that the reader have permission to use these tools on his computer system or network. Use of these tools and the analysis of virulent software always carry some inherent risk that must be securely managed and adequately mitigated.

An in-depth study of memory analysis techniques is outside the scope of this work, as it requires a comprehensive study of operating system internals and software reverse engineering techniques. Instead, this work should be considered as a guide to using the Volatility memory analysis framework for the analysis of a Linux-based memory malware infection.

In this report, the use of the words rootkit, infection and malware are used interchangeably. The same applies for kernel module, driver and Loadable Kernel Modules (LKM).

Finally, the use of masculine is employed throughout this text for the purpose of simplification.

Availability of Linux memory images and profiles

Various Linux-based memory images are available from different publicly available sources, most notably among them those from SecondLook. The author, for the time being, has endeavoured to build his own virtual machines and memory profiles to be independent of those already available.

The author will endeavour to ensure that his memory images and profiles will be made available to anyone requesting a copy, given the constraints below. The author can be contacted at val-forensics@drdc-rddc.gc.ca. Please state your name, organization, country and mailing address including additional contact information and one will be mailed to you within a reasonable delay. No PO Boxes will be accepted—commercial and government mailing addresses only.

However, countries listed on Canada's Export Control List (ECL) are automatically forfeit. Even though the various memory images and Volatility profiles are considered as NON-CONTROLLED GOODS and are for unclassified use and unlimited distribution, ECL-listed countries will NOT be considered. For more information concerning countries listed on Canada's ECL, please consult http://www.international.gc.ca/controls-controles/export-exportation/exp_ctr_handbook-manuel_ctr_exp-p2.aspx?lang=eng#d.

1 Background

1.1 Objective

The objective of this report is to examine how a computer forensic investigator/incident handler, without specialised computer memory or software reverse engineering skills, can successfully investigate a Linux-based memory image suspected of infection.

To successfully investigate such an image, this report describes an applied plugin-based approach as it uses demonstrable procedures that intermediate-level investigators and incident handlers can apply as a basis for investigating suspected memory images.

The work is based on the publicly available source code for the KBeast rootkit. This document is the first in a series of reports that examines Linux-based malware memory analysis. This specific report surveys what to look for when examining a kernel-based rootkit. Ultimately, these reports will provide a foundational framework that novice and experienced investigators alike can rely on for guidance when investigating infected Linux memory images.

Unlike the previous Windows-based reports, it was determined that Linux-specific memory analysis case studies and reports have not been examined by the community, at least as of the time of this writing, hence prompting the author to write this case study and its subsequent follow-up studies.

1.2 Project support

This work was carried out over a period of several months as a collaborative effort between DRDC – Valcartier Research Centre and the RCMP, as part of the Live Computer Forensics project (SRE-09-015, 31XF20).

1.3 Target audience

The results of this project may also be of significant interest to the Canadian Forces Network Operations Centre (CFNOC), the RCMP's Integrated Technological Crime Unit (ITCU), the Sûreté du Québec and other law enforcement-related cyber investigation teams.

The target audience for this report is the computer forensic investigator who assesses suspect computer memory images for evidence of infection and the incident handler who is called on to assess or intervene in a possible malware infection. While previous reports were targeted at investigators and incident handlers working with Windows-based memory images and malware, this new series of reports will be directed at those who must analyse Linux malware-infected memory images.

The skills amassed by incident handlers and investigators alike while using Volatility to examine Windows memory images will be of some help. However, Linux and Windows are not the same and while there is commonality in the approach used by the author throughout both series of

reports, important differences are apparent. To extract the maximum value from this report, the reader should have a working knowledge of Linux, basic system administration and software compilation.

1.4 KBeast rootkit background

After extensive web searches, no technical analysis could be found from the various anti-virus (AV) vendors concerning the KBeast rootkit. What is mostly known about the capabilities of IPSECS KBeastv1 can be identified from reading its source code. The rootkit was written December 2011 by Ph03n1x of IPSECS (http://ipsecs.org), as based on the information found in its source code.

However, a 2012 Volatility analysis of KBeast was written by Andrew Case [14]; while it makes for an interesting read, it was not used as the basis of this report as it goes into reverse engineering efforts. This current paper, however, provides a more detailed Volatility-based analysis but without the need to understand reverse engineering. Additional information about KBeast can be found from [15, 16], although the latter provides information with respect to hypervisor-level detection of rootkits.

KBeast is a LKM rootkit that is compiled using the provided shell script *setup* and then loaded into kernel space. However, in order for the module to be loaded into kernel space via the *setup* script, the user/attacker must already have root-level permission.

According to the rootkit's author, the rootkit has the following capabilities [1]:

- ability to hide itself;
- ability to hide files and directories;
- ability to hide processes (e.g., ps, pstree, top, lsof);
- ability to hide sockets and connections (e.g., netstat, lsof);
- keystroke logging to capture user activity;[1]
- anti-killing of processes;
- anti-removal of files;
- anti-deletion of this rootkit;
- local root escalation backdoor; and
- hidden remote backdoor binding.

However, many of these capabilities must be configured in the rootkit's configuration file, *config.h* [1, 2]. As for the compilation of this rootkit, however, the author made no changes to its configuration and instead accepted the rootkit's default setup.

[1] Source code file, *config.h*, indicates that this capability may make the rootkit unstable. However, this is the default behavior for this rootkit.

While a brief analysis of the source code by the author indicates that these capabilities appear to be valid claims, the author has not verified these capabilities in-depth. Moreover, the rootkit's author claims that it has been tested on kernels 2.6.9 up to 2.6.32, and that it may work against more recent kernels [1]. Compilation specifics are found in Section 1.7.

1.5 Information concerning the guest VM

The Linux test virtual machine (VM) that was to be infected with KBeast was built atop Ubuntu 10.10 x86 and was installed from DVD media. The VM was allocated 1 CPU and 2 GiB RAM and the default Ubuntu VirtualBox parameters for the VM were used. Once the VM's operating system was installed and functional, VirtualBox's Guest Additions were installed therein. The system appeared to be in good working order except that *dwarfdump* and its required dependencies were not installed from the media during installation and the online repositories for Ubuntu 10.Ten were no longer available. Thus, the source code for the variously required packages had to be downloaded from the web, compiled and then installed within the VM. Once this was done, the operating system was then temporarily shut down.

1.6 Compiling and loading the rootkit

The rootkit's source code, found in downloaded file *ipsecs-kbeast-v1.tar.gz* (SHA1 hash of 9390A90C0689D272DDA16CE924CF18B24C41DF9D), was copied over to the VM through the mounting of a *Shared Folder* (mounted read-only) to directory */tmp,* where it was unpackaged and compiled according to the following commands:

```
$       tar xzf ipsecs-kbeast-v1.tar.gz

$       cd kbeast-v1

$       ./setup build 1
```

Upon successful compilation, the rootkit is then loaded into kernel space and the default rootkit location, */usr/_h4x_*, as specified in the rootkit's configuration file *config.h,* is created. The rootkit, *_h4x_bd*, is then copied therein and the keylogging file *acctlog.0* is created.

Finally, although configuration changes can be made to *config.h*, which stores the user (or attacker – depending on the case) configurable settings, none were made during this infection.

1.7 Memory image metadata

Two memory images were taken of the VM. One was taken just prior to infection and the other just after rootkit infection. In so doing, it is possible to compare a clean system to an infected system in the event that such comparative information is required during the analysis of the infected memory image.

Similarities in the fuzzy hashes for these two memory images have been identified in the tables below (found as pink characters in tables 1 and 2) to identify large memory structures that have more or less remained the same [5].

Both acquired memory images should have been exactly 2 GiB in size, but as it turned out were not. Instead, they were each approximately 6.4% larger, thereby indicating that the VirtualBox-specific overhead for this memory dump was non-negligible.

The VM's memory was dumped to obtain an uninfected baseline memory. This was done by restarting the VM using the following command [9]:

```
$ virtualbox --debug --startvm "Ubuntu 10.10 x86"
```

VM memory was then dumped using the following command [9]:

```
$ vboxmanage debugvm "Ubuntu 10.10 x86" dumpguestcore --filename
ubuntu10_10_kbeast.mem
```

This process was repeated shortly after infection of the VM.

The Volatility profile, *ubuntu_10_10_profile.zip*, was generated as per the instructions found in [9]. The profile is available to the reader as per the eligibility requirements set out on *page xiii*.

1.7.1 Uninfected baseline memory image metadata

The following metadata accurately describes the uninfected baseline memory image:

Table 1: Linux Ubuntu 10.10 x86 uninfected memory image metadata.

Memory image name	Ubuntu_10_10.mem
Actual size (exact)	2,285,979,432 bytes
Expected size (exact)	2,147,483,648 bytes
SHA1 hash	7f557359ee9c64b90a32f72cbd575b1de4e81107
Fuzzy hash	6291456:hYw8PWLUj+Q/TFUHvsfw+yVH67T8nU7Rp+0wgYjtVw0 8:PWnNj

1.7.2 Infected memory image metadata

The following metadata accurately describes the infected memory image:

Table 2: Linux Ubuntu 10.10 x86 KBeast infected memory image metadata.

Memory image name	ubuntu_10_10_kbeast.mem
Actual size (exact)	2,285,979,432 bytes
Expected size (exact)	2,147,483,648 bytes
SHA1 hash	a7838790933fe822012ed11006d7adb0eb070279
Fuzzy hash	6291456:zp6c1s8g8h+Q/TFUH98iw+hkVZ7YlRIPnUwkZjYMbkVw0Y:MWcIm7

1.8 AV scanners used

This report makes use of six anti-virus scanners, the same six as those used in reports [6, 7]. These scanners continue to represent a wide cross-section of various detection mechanisms necessary for the identification of diverse malware. Each scanner was updated July 23, 2014; the analysis was then carried out. Scanner specifics are listed in the following table:

Table 3: List of anti-virus scanners and their command line parameters.

Anti-virus scanner	Command line parameters
Avast v.1.3.0 command line scanner	avast -c
AVG 2013 command line scanner version 13.0.3114	avgscan -H -P -p
BitDefender for Unices v7.90123 Linux-amd64 scanner command line	bdscan (no parameters used)
Comodo Antivirus Product Version 1.1.268025.1 / Virus Signature Database Version 16954	cmdscan -v -s
FRISK F-Prot version 6.3.3.5015 command line scanner	fpscan -u 4 -s 4 -z 10 --adware --applications --nospin
McAfee VirusScan for Linux64 Version 6.0.3.356 command line scanner	uvscan --RECURSIVE --ANALYZE --MANALYZE --MIME --PANALYZE --UNZIP --VERBOSE

2 Peripheral concerns

2.1 Why examine Linux memory images or make them available?

After extensively searching the available public literature, it became clear that few detailed Linux-based memory analyses could be found. In addition, those few reports or documents that were found were not of sufficient quality to enable others to readily learn the necessary techniques or approaches to conducting their own analyses that were specifically targeted towards non-memory specialists and non-reverse engineers. The author has opted to build his own virtual machines and infect them to be independent of those already done.

The author asserts that by methodically conducting various Linux-based memory analyses using a memory analysis framework such as Volatility and sharing the techniques and methods used for these analyses with the digital forensics community, it will help to further advance the capabilities of investigators and incident handlers alike when dealing with potentially infected Linux memory images. Just as with the now completed Windows series of reports, which provide a detailed methodology for conducting Volatility-based malware memory analysis for non-experts, this series of Linux-based reports hope to have the same impact for the Linux audience.

2.2 Volatility background

The 2.4 version of the framework is considered the stable public release and is suitable for use by both the general public and investigators alike, although it may not necessarily have the most recent or bleeding-edge plugins. It was released for public use August 2014.

Originally written by Aaron Walters of Volatile Systems, Volatility has become a full-fledged memory analysis framework. It is written entirely in Python and can therefore be run atop Windows, Linux and other various operating systems supporting Python. Volatility began supporting Linux-based memory analysis in previous versions, although its current support has improved a great deal. However, its Windows support continues to remain both more robust and reliable. Currently, it is developed by a variety of contributors, although the most well-known of these are Michael Ligh, Jamie Levy, Brendan Dolan-Gavitt, Andrew Case and Mike Auty. Furthermore, each of these individuals has made significant contributions to the digital forensics community over the last few years. Michael Cohen, who was formerly with the project, has gone on to found *Rekall* (https://code.google.com/p/rekall), a memory analysis framework similar to Volatility that at the time of this writing is not yet ready for public use.

The Linux plugins supported by version 2.4 of Volatility are described in Annex A.

2.3 Purpose of these tutorials

Although online tutorials concerning infected Linux-based memory images exist, these tutorials are generally written for a highly technical audience already familiar with software reverse engineering and memory forensics. They typically provide either too little information or are too technical to be of much use to most investigators and incident handlers.

Thus, the author asserts that re-examining and thoroughly documenting the steps and procedures used to identify various rootkit-based infections will aid the reader in unravelling his own malware-based investigations. It is hoped that these reports will build a compendium of knowledge to serve the forensic community as learning guides and tutorials.

The author has made all efforts to ensure that this document and the investigation of the KBeast rootkit are comprehensible to the general computer forensic practitioner, in the hopes of reaching as wide an audience as possible and having a more significant impact.

2.4 Issues concerning data carving

Unlike for Windows-based memory images, data carving is not particularly effective against Linux-based memory images. Experimentation by the author has revealed that once a Linux binary, whether an executable or a compiled library file, has been loaded into memory, it loses its ELF header, thereby making its detection and subsequent carving very difficult. Without an ELF header from which to start, data carvers and recovery software will not be able to identify the starting point of a given library or executable in memory. The author attempted to perform data carving using the same tool with consistent settings (Photorec) against ten different memory images obtained against both 32 and 64-bit Linux operating systems. Between them, only one ELF-based file was ever recovered. The other files recovered were mostly text-based data files.

The reader may recall that these same data carving techniques worked moderately well against Windows-based memory images. This is because Windows executables and libraries have their PE header loaded into memory, making them readily identifiable and recoverable.

Moreover, the various techniques examined in the Windows series of reports found that occasionally some of the malware carved from a memory image matched those dumped from the memory image using Volatility. What this means is that data recovery tools and software are more likely to recover intact (or partially intact) malware from Windows memory images as compared to those from Linux. The various MD5/SHA1 and fuzzy hashing (file similarity matching) used for Windows also confirms this assertion. As of Volatility 2.4, a new plugin, *linux_elf*, has been designed to help investigators determine where ELF files are residing within a memory image using alternate means.

2.5 Issues concerning AV analysis

Further complicating Linux-based malware memory analysis is the lack of Linux-specific malware detection using various AV scanners. While the various scanners used throughout the Windows reports worked well against both Volatility-dumped and data-carved files, these very

same AV scanners (Avast, AVG, BitDefender, ClamAV,[2] Comodo2, Frisk F-Prot and McAfee) fared poorly against the Linux-based rootkits. Quite the opposite was in fact expected. Since these rootkits were all open source, it would have followed that the various scanners would have included some basic signature or heuristic detection capability. After all, these rootkits will inevitably be used as the basis for future rootkits. Unfortunately, this was not the case at all.

Thus, both this report and the series of Linux-based reports will make little use of AV scanners. That, however, requires the reader to have a very good understanding of Linux to make up for what the scanners fail to detect. Nevertheless, certain portions of each Linux-based report will still use AV scanners in the hope that they may be able to reveal something pertinent concerning a rootkit. Specifics are available in the analysis portion of this and subsequent follow-up reports.

2.6 Issues concerning the NSRL

The National Software Reference Library (NSRL) is a standardised and trustworthy source of computer operating system and application file names and hashes (MD5/SHA1). It is not particularly well suited to Linux-based investigations as there are far too many Linux distributions (hundreds of publicly available distributions are known to exist) to be covered by the NSRL, including all the various kernel versions in use.[3] As such, it does not make sense to rely on the NSRL for file name listings and hashes for comparative purposes against data files recovered from a Linux memory image. For that reason, these reports and their examination of various infected Linux memory images will not use the NSRL as was done for the Windows series of reports.

[2] This AV was used in some Windows memory malware reports but not others.
[3] A full listing of which Linux distributions are supported by a given version of the NSRL can be found in its "*nsrlprod.txt*" file.

3 Memory investigation and analysis of KBeast using Volatility

This section has two main parts. The first, in Section 3.1, is short and quickly determines if the compiled rootkit and its source code are identifiable using the aforementioned AV scanners (Section 1.8) while the second, Section 3.2, performs the actual Volatility analysis.

3.1 Step 1: AV analysis of memory images and source code

This step examines an infected memory image, source code and compiled rootkit using the various scanners in the hope of identifying any of them as infected.

3.1.1 Memory image analysis

None of the scanners listed in Section 1.8 found anything in memory image *ubtunu_10_10_kbeast.mem*.

3.1.2 Source code analysis

The rootkit's source code was then analysed using the aforementioned scanners. Upon scanning the source code files, nothing was identified as malicious or infected by any of the scanners.

3.1.3 Rootkit analysis

The compiled rootkit, file *_h4x_bd*, was obtained by manually mounting the VM disk image and copying it to the host system's disk. This file was then scanned using the aforementioned scanners where nothing was detected.

This file is 11,998 bytes in size with a SHA1 hash of EC1FE622BD57A73369E70741D F3540EC467269CA. It was submitted to VirusTotal[4] for inspection against a total of 51 scanners, all of which failed to detect anything.

3.2 Step 2: Volatility system information extraction

This next step examines the infected memory image using Volatility plugins that provide system information about the suspect computer and its operating system. To understand the rootkit itself its source code must be thoroughly understood, which is beyond the goal of this paper. However, to better understand its capabilities, a thorough analysis of it using Volatility will help the reader to better understand it.

[4] More information concerning the submission of this particular rootkit can be found at https://www.virustotal.com/en/file/0ad488a33e46fe317807de834a302d9b6685803a9a8760c932f5de948a15c7f9/analysis/. (Access date: 1 Aug 2014).

3.2.1 Plugin linux_banner

This plugin is used to determine the Linux kernel, its revision and architecture. The plugin was run using the following command:

```
$       volatility --profile=Linuxubuntu_10_10_profilex86 -f
        ubuntu_10_10_kbeast.mem linux_banner
```

The plugin then generated the following output:

```
Linux version 2.6.35-22-generic (buildd@rothera) (gcc version 4.4.5
(Ubuntu/Linaro 4.4.4-14ubuntu4) ) #33-Ubuntu SMP Sun Sep 19 20:34:50 UTC
2010 (Ubuntu 2.6.35-22.33-generic 2.6.35.4)
```

The output indicates that Linux 2.6.35-22 is running and that it is an SMP-enabled kernel, compiled using GCC version 4.4.5 (September 19, 2010). However, looking only at this information it is not possible to determine if it is a 32-bit, 32-bit PAE or 64-bit kernel. To be fair, part of the problem is the kernel naming convention used by Debian and Ubuntu, in this case, for example, Ubuntu 2.6.35-22.23-generic 2.6.35.4.

3.2.2 Plugin linux_cpuinfo

This plugin is used to identify the type and number of CPUs running atop the suspect computer. The plugin was run using the following command resulting:

```
$       volatility --profile=Linuxubuntu_10_10_profilex86 -f
        ubuntu_10_10_kbeast.mem linux_cpuinfo
```

Processor	Vendor	Model	
0	GenuineIntel	Intel(R) Xeon(R) CPU	X5355 @ 2.66GHz

The make and model of the two identified processors are correct. The base processor speed is 2.66 GHz.

3.2.3 Plugin linux_dmesg

This plugin is used to identify important boot-up information and kernel-based messages about the underlying computer system. The UNIX/Linux *dmesg* command, upon which this plugin is based, identifies various kernel and device driver boot-up information and output structures found residing in memory that are typically found in system file */var/log/dmesg.*[5]

Using this plugin, it may be possible to identify what kernel (and its version) was running, the number and type of CPUs, instantiated system services, the map of system memory, networking

[5] Not all UNIX systems necessarily use this specific file. Mileage will vary according to the underlying operating system.

and many other essential capabilities (both software and hardware) that a typical Linux system will have. The plugin was run using the following command:

```
$       volatility --profile=Linuxubuntu_10_10_profilex86 -f
        ubuntu_10_10_kbeast.mem linux_dmesg
```

The output is too long to list here, but a full listing can be found in Annex B.1. After a detailed inspection of the output, nothing out of the ordinary was identified.

3.2.4 Plugin linux_iomem

This plugin provides the physical memory mapping of the suspect computer system, which in this case is a virtual machine. An in-depth examination of this virtual machine's physical memory mapping is outside the scope of this report; however, additional information concerning the interpretation of this data can be found in [4].

The plugin was run using the following command:

```
$       volatility --profile=Linuxubuntu_10_10_profilex86 -f
        ubuntu_10_10_kbeast.mem linux_iomem
```

The output of this plugin is listed in the first three columns of Table 4. The fourth and fifth columns were added by the author to facilitate reading. In blue, we find the virtualized hardware RAM (equivalent to computer hardware memory modules).

Table 4: VM physical memory mapping for suspected system.

Hardware	Starting Address	Ending Address	Size Difference	Size (in bytes)
PCI mem	**0x0**	**0xFFFFFFFF**	**FFFFFFFF**	**4,294,967,296**
reserved	0x0	0xFFF	FFF	4,096
System RAM	**0x1000**	**0x1FFF**	**FFF**	**4,096**
reserved	0x2000	0xFFFF	DFFF	57,344
System RAM	**0x10000**	**0x9FBFF**	**8FBFF**	**588,800**
reserved	0x9FC00	0x9FFFF	3FF	1,024
Video RAM area	0xA0000	0xBFFFF	1FFFF	131,072
Video ROM	0xC0000	0xC7FFF	7FFF	32,768
Adapter ROM	0xE2000	0xE2FFF	FFF	4,096
reserved	0xF0000	0xFFFFF	FFFF	65,536
System ROM	0xF0000	0xFFFFF	FFFF	65,536
System RAM	**0x100000**	**0x7FFEFFFF**	**7FEEFFFF**	**2,146,369,536**
Kernel code	0x100000	0x5D029D	4D029D	5,046,942
Kernel data	0x5D029E	0x818667	2483C9	2,393,034
Kernel bss	0x8CA000	0x9A0ADB	D6ADB	879,324
ACPI Tables	0x7FFF0000	0x7FFFFFFF	FFFF	65,536

Hardware	Starting Address	Ending Address	Size Difference	Size (in bytes)
0000:00:02.0	0xE0000000	0xE7FFFFFF	7FFFFFF	134,217,728
0000:00:03.0	0xF0000000	0xF001FFFF	1FFFF	131,072
e1000	0xF0000000	0xF001FFFF	1FFFF	131,072
0000:00:04.0	0xF0400000	0xF07FFFFF	3FFFFF	4,194,304
vboxguest	0xF0400000	0xF07FFFFF	3FFFFF	4,194,304
0000:00:04.0	0xF0800000	0xF0803FFF	3FFF	16,384
0000:00:06.0	0xF0804000	0xF0804FFF	FFF	4,096
ohci_hcd	0xF0804000	0xF0804FFF	FFF	4,096
0000:00:0b.0	0xF0805000	0xF0805FFF	FFF	4,096
ehci_hcd	0xF0805000	0xF0805FFF	FFF	4,096
0000:00:0d.0	0xF0806000	0xF0807FFF	1FFF	8,192
ahci	0xF0806000	0xF0807FFF	1FFF	16,384
Local APIC	0xFEE00000	0xFEE00FFF	FFF	4,096
reserved	0xFFFC0000	0xFFFFFFFF	3FFFF	262,144

PCI mem indicates that this 32-bit system can address a maximum of 4 GiB (4,294,967,296 bytes) of memory, without the use of PAE, which this kernel does not support.

The VM, allocated a total of 2 GiB (2,147,483,648 bytes) is able to use 2,146,962,432 bytes, leaving 521,216 (509 KiB) bytes hidden to the VM's operating system. This hidden memory was used and reserved by the VM's BIOS.

The reason an investigator/incident handler should use this plugin is to be aware of the different address ranges used by the hardware (virtualized or not) and operating system. This information can be used to validate that the malware has not tricked the operating system's virtual memory manager or other kernel components into thinking the system has less memory than it physically has. Had the amount of unseen memory been significantly larger than the 509 KiB used by the BIOS, then this could have indicated that the malware was busy making changes to the system to hide itself. While this capability has not yet been seen in Linux malware, this does not preclude it from existing.

3.2.5 Plugin linux_slabinfo

Plugin *linux_slabinfo* is used to provide kernel SLAB-based information. The kernel SLAB structure is a specific structure kept in */proc* used to keep track of the different kernel structures that rely on various caches. These include, but are not limited to filesystem buffers, network buffers and caches, inodes and many others.

This plugin only supports SLAB-based kernels and as such will only work with memory images using kernel 2.6.22 and earlier. Kernels 2.6.23 and later, by default, use SLUB-based memory management [11, 12, 13].

3.2.6 Plugin linux_mount_cache

Plugin *linux_mount_cache* is used to provide kernel SLAB-based information. The kernel SLAB structure is a specific structure kept in */proc* used to keep track of different kernel structures that rely on various caches. These include, but are not limited to, filesystem buffers, network buffers and caches, inodes and many others.

The reason this plugin does not work is that it supports SLAB-only based kernels, not SLUB-based kernels [11, 12, 13].

3.2.7 Plugin linux_mount

Although this plugin is not the preferred manner for obtaining a list of mounted disk, kernel and virtual filesystems, it did work, unlike the previous plugin, even if some of the output is not the same as what the *linux_mount_cache* plugin would produce.

The plugin was run using the following command generating the following output:

```
$ volatility --profile=Linuxubuntu_10_10_profilex86 -f
ubuntu_10_10_kbeast.mem linux_mount
```

This plugin then resulted in the following output, which has been reformatted and whose order has been rearranged to improve its legibility:

- /dev/disk/by-uuid/b13dedba-11eb-497f-96b2-e06d37b3aef1/ ext4 rw,relatime
- binfmt_misc /proc/sys/fs/binfmt_misc binfmt_misc rw,relatime,nosuid,nodev,noexec
- fusectl /sys/fs/fuse/connections fusectl rw,relatime
- gvfs-fuse-daemon /home/richard/.gvfs fuse rw,relatime,nosuid,nodev
- none /dev devtmpfs rw,relatime
- none /dev/pts devpts rw,relatime,nosuid,noexec
- none /dev/shm tmpfs rw,relatime,nosuid,nodev
- none /proc proc rw,relatime,nosuid,nodev,noexec
- none /sys sysfs rw,relatime,nosuid,nodev,noexec
- none /sys/kernel/debug debugfs rw,relatime
- none /sys/kernel/security securityfs rw,relatime
- none /var/lock tmpfs rw,relatime,nosuid,nodev,noexec
- none /var/run tmpfs rw,relatime,nosuid

Upon closer examination of the output, nothing appeared out of the ordinary. The above list is what would be expected from a typical modern Linux desktop.

3.2.8 Summary

Performing Volatility system information extraction has demonstrated that collecting information about the VM's underlying operating system and base configuration is straightforward. However, despite the many pages of output generated by the various plugins, no clues or hints as to this memory image's infection could be identified.

These plugins do provide important basic information about the underlying hardware and operating system, which, while informative, are unlikely to yield immediate clues. Upon correlation with additional plugins (yet to be used), they may yield further information.

The author is of the opinion that the most important plugin in this step is *linux_dmesg*. However, plugins *linux_iomem* and *linux_mount* may provide additional indications of malware presence, but only if the malware is capable of modifying the kernel's perception of available "System RAM" or mount points, respectively. Plugin *linux_banner* is useful for obtaining information about the version of the kernel in use but on its own provides no information about the architecture of the kernel (32 or 64-bit).

It is important that analysts use the appropriate Volatility plugins supported by the memory image's underlying kernel and recognize which is SLUB and SLAB based. That is the reason why the author continues to use them even though they will not work against more recent versions of the Linux kernel.

3.3 Step 3: Volatility process listings and analysis

In this step, specific plugins will be used to identify process-based information concerning the infected memory image.

3.3.1 Plugin linux_psaux

This plugin is used to provide a full process listing of the system. Its output is approximately the same as would be obtained running the *ps -aux* command via a terminal. The plugin was run using the following command:

```
$       volatility --profile=Linuxubuntu_10_10_profilex86 -f
        ubuntu_10_10_kbeast.mem linux_psaux
```

The resulting output consisted of 134 listed processes. However, as this output is too long to place in this section, it can instead be found in Annex B.2.

Everything in this list of processes appeared to be normal, with the exception of one process, specifically the very last process in the list. Its name of *_h4x_bd* is not a known standard Linux

process. Running with a PID of 1980 and a UID/GID of 2, this process may be running with administrative privilege[6]. Thus, more attention is warranted with respect to this process.

3.3.2 Plugin linux_pslist

This is an interesting Volatility plugin as it too is used to list all running processes on a system. It works by walking the *task_struct->tasks* linked list [10], similar to Volatility's Windows process listing plugins, except only for Linux. The plugin can list all active processes (except for the system swapping process(es)) [10]. According to Volatility's documentation [10], if the output under the DTB column is blank then it is very likely a kernel thread. This includes drivers and other kernel modules visible from userland.

The plugin was run using the following command:

```
$ volatility --profile=Linuxubuntu_10_10_profilex86 -f
  ubuntu_10_10_kbeast.mem linux_pslist
```

The resulting output is too long to include here but can be found in Annex B.3. Importantly, the same numbers of processes (134) were found using this plugin as with the previous plugin. Again, the only process that did not appear to belong was *_h4x_bd*.

3.3.3 Plugin linux_pslist_cache

This plugin attempts to build a list of active processes from *kmem_cache*, the kernel's memory cache [10]. In effect, it should reproduce the same results as the *linux_pslist* plugin using a different mechanism, which is useful in corroborating the results of the other available process listing plugins.

The reason this plugin does not work is that it supports SLAB-only based kernels, not SLUB-based kernels [11, 12, 13].

3.3.4 Plugin linux_pstree

The purpose of this plugin is to identify the relationship between processes, in effect to identify a given process' parent (or PPID). The reader may have noticed that to date none of the Linux process listing plugins provides the PPID of the variously identified processes. Thus, to identify these relationships, the following command was issued:

```
$ volatility --profile=Linuxubuntu_10_10_profilex86 -f
  ubuntu_10_10_kbeast.mem linux_pstree
```

[6] UID/GID 0 typically denotes *root*, while UID/GID 1 usually points to *bin* and a UID/GID of 2 generally implies *daemon*. Mileage will vary for all non-root accounts (root always has UID/GID of 0) for standard UNIX and Linux systems.

Again, this plugin identifies 134 processes. Suspicious process *_h4x_bd* (PID 1980) does not appear to have been instantiated by any other process, indicating that it was either launched by *init* or by the kernel. Finally, the plugin indicates that the process had a UID of 2. The output of this plugin, too long to be listed herein, can be found in Annex B.4.

3.3.5 Plugin linux_pidhashtable

This interesting plugin can be used to identify hidden or previously unseen processes. However, it is not the same as the Windows *psxview* plugin. Instead, it works by walking the PID hash table [10]. The *plugin* validates that a given process forms part of the PID hash table maintained by the operating system. This lookup (or hash) table is similar to that used by Windows in that they are both doubly linked lists. In the same manner that rogue Windows processes can unlink themselves from the Windows process table, rogue Linux processes can unlink themselves from the PID hash table and this plugin can aid in identifying them. Its output is not that different from the *linux_pslist* plugin.

The plugin was run using the following command:

```
$ volatility --profile=Linuxubuntu_10_10_profilex86 -f
ubuntu_10_10_kbeast.mem linux_pidhashtable
```

The plugin's output is too long to list here but can be found in Annex B.5. Looking at its output, 256 processes in all, nothing appears to be out of place. Many more items are listed than with the previous process listing plugins, but ultimately nothing stands out other than the previously mentioned suspicious process *_h4x_bd* (PID 1980).

3.3.6 Plugin linux_psxview

This plugin is similar to the *psxview* plugin used for Windows memory investigations. However, this Linux-specific plugin makes use of the very different data structures found in Linux-based memory images.

Memory offsets are specified in terms of virtual addresses. Rather than support five distinct memory analysing algorithms, this particular plugin uses only three. The first of these is *pslist*, which uses the same technique used by the *pslist* plugin (see Section 3.3.2 for details). The second is *pid-hash*, which helps identify hidden processes (see Section 3.3.5 for details). The third is *kmem_cache*, which examines the kernel's memory cache (see Section 3.3.3). Specifically, this cache stores information about not only ongoing processes but also metadata concerning terminated processes, sometimes even those that may have finished long ago, depending on the degree of process creation within the operating system [10].

The plugin was run using the following command:

```
$ volatility --profile=Linuxubuntu_10_10_profilex86 -f
ubuntu_10_10_kbeast.mem linux_psxview
```

The output from the plugin is far too long to list here but can be found in Annex B.6. Looking at the list, 256 processes in all, the very same processes were identified as by plugin *linux_pidhashtable* (see Section 3.3.5 for details). Upon closer examination, nothing appeared out of the ordinary with this plugin's output other than process *_h4x_bd* (PID 1980).

3.3.7 Summary

Performing Volatility process listings and analysis has shown that four issues are noteworthy. The first is that process *_h4x_bd* (PID 1980) is suspicious in that it does not fit with the overall nomenclature of the variously identified processes as per the aforementioned plugins. However, there is currently no other useful information to go on.

The second issue is that a normal Linux system can have many processes running that are not necessarily found using a standard process listing (plugins *linux_psaux*, *linux_pslist* and *linux_pslist_cache*).

Thirdly, process 1980 does not have a parent process indicating it was likely instantiated either by the kernel or by *init*.

The final issue is that a process running with a UID/GID of 2 is neither a user process nor a root process, although this will depend on the specifics of each system's individual */etc/passwd* and */etc/group* files which determine the details of UID and GID. On many systems, Linux and UNIX alike, a UID/GID of 2 typically indicate an administrative account.

3.4 Step 4: Volatility history listing

In this step, various command shell listing plugins will be used to attempt to identify pertinent shell histories.

3.4.1 Plugin linux_bash

This particular plugin searches a memory image for command shell histories, similar to Volatility's Windows-based command history plugins. This is a brute force plugin in that it scans the entire memory image for signs of shell histories and as such may output erroneous information [10].

The plugin was run using the following command:

```
$       volatility --profile=Linuxubuntu_10_10_profilex86 -f
        ubuntu_10_10_kbeast.mem linux_bash
```

Running with the *-A* option can help ensure that all processes are scanned for shell history information, but it can take many hours to process a large memory image. It is also possible that the plugin will crash when used with this option. However, the option is useful because attackers may have copied the shell program (i.e., *bash*) to another name (i.e., */tmp/hsab*) and ran it to

circumvent the manual detection of shell histories in memory. Tests have found that the same amount of information was identified when the plugin was used without the option. Of course, mileage will vary.

In a typical investigation, there could be hundreds or even thousands of lines of shell history to go over. Typically, after a system reboot, pre-existing shell histories will no longer be recoverable from memory; this, of course, is only a rule of thumb and there are times when pre-reboot data will remain intact in memory for recovery.

Relevant output generated by the plugin is listed in Table 5 detailing the compilation of the KBeast rootkit by the author.

Table 5: Plugin output for linux_bash (pruned and sorted chronologically).

PID	Name	Command Time	Command
1521	bash	2014-05-23 17:42:27 UTC+0000	su - root
		…	
1550	bash	2014-05-23 17:43:07 UTC+0000	uz ipsecs-kbeast-v1.tar.gz
1550	bash	2014-05-23 17:43:11 UTC+0000	cd kbeast-v1/
1550	bash	2014-05-23 17:43:12 UTC+0000	ls
1550	bash	2014-05-23 17:43:13 UTC+0000	ls -al
1550	bash	2014-05-23 17:43:19 UTC+0000	less README.TXT
1550	bash	2014-05-23 17:43:40 UTC+0000	ls -al init/
1550	bash	2014-05-23 17:43:48 UTC+0000	cat init/initcheck.txt
1550	bash	2014-05-23 17:44:01 UTC+0000	cat init/sysstat
1550	bash	2014-05-23 17:44:16 UTC+0000	ls -al init/
1550	bash	2014-05-23 17:44:26 UTC+0000	cat init/sysstat.diff
1550	bash	2014-05-23 17:44:33 UTC+0000	cat init/sysstat.orig
1550	bash	2014-05-23 17:44:36 UTC+0000	ls -al
1550	bash	2014-05-23 17:44:43 UTC+0000	./setup build

Note that PID 1521 was the currently logged in user's bash shell while PID 1550 is a new shell that is directly attributable to that user having successfully *su'ed*. However, an attacker's shell command histories will not be retrievable in every case.

3.4.2 Summary

Performing a Volatility shell history listing has demonstrated that command histories can produce rewards, especially if a system's memory is acquired within several hours of compromise (or possibly longer than that if the system is quiescent).

However, both what is recovered and its pertinence can vary greatly between investigations and, as such, investigators and incident handlers must remember that such plugins represent only one small piece of the analysis. These plugins rely on the *bash* shell, which is not always the system or user default shell; thus, these plugins do have their limits.

3.5 Step 5: Volatility file detection and dumping

In this step, various plugins will be used to attempt to isolate and dump important or suspicious files for further analysis.

3.5.1 Plugin linux_lsof

This plugin lists all open files, sockets, pipes, directories and other objects that the system currently has open for a given process, a list of processes, or all processes. This plugin functions similarly to the UNIX/Linux command *lsof*; however, it does not in any way list the same number of files or details as the real *lsof* command does. For example, a typical Linux system running with X Windows will have at least several thousand more open filesystem objects[7]. However, in using this plugin, it is likely that less than half of these will actually be open.

The plugin was run using the following command:

```
$       volatility --profile=Linuxubuntu_10_10_profilex86 -f
        ubuntu_10_10_kbeast.mem linux_lsof -p 1980 -v
```

Relevant output generated by the plugin is listed in Table 6.

Table 6: Plugin output for linux_lsof (sorted by FD).

PID	FD (File Descriptor)	Path
1980	1	/dev/pts/0
1980	2	/dev/pts/0
1980	3	socket:[12342]

Although no specific file name for suspicious process 1980 could be identified using this plugin, it succeeded in detecting socket 12342, which further augments the suspiciousness of this process.

3.5.2 Plugin linux_proc_maps

This very powerful plugin can be used to learn important information about the underlying system as a whole or about one or more specific processes. Specifically, this plugin is used to identify process metadata including the name and location of running processes, shared libraries, stacks, inodes and memory address ranges.

[7] This test was carried out on the infected VM after memory acquisition using the command "*lsof | wc -l*".

The plugin was run using the following command (against suspicious process 1980):

```
$       volatility --profile=Linuxubuntu_10_10_profilex86 -f
        ubuntu_10_10_kbeast.mem linux_proc_maps -p 1980
```

Relevant output generated by the plugin is listed in Table 7.

Table 7: Plugin output for linux_proc_maps (sorted by start address).

PID	Start	End	Flags	Pgoff	Major	Minor	Inode	File Path
1980	0x0000000000827000	0x0000000000828000	r-x	0x0	0	0	0	
1980	0x0000000000ac7000	0x0000000000ae3000	r-x	0x0	8	1	1572886	/lib/ld-2.12.1.so
1980	0x0000000000ae3000	0x0000000000ae4000	r--	0x1b000	8	1	1572886	/lib/ld-2.12.1.so
1980	0x0000000000ae4000	0x0000000000ae5000	rw-	0x1c000	8	1	1572886	/lib/ld-2.12.1.so
1980	0x0000000000c45000	0x0000000000d9c000	r-x	0x0	8	1	1572910	/lib/libc-2.12.1.so
1980	0x0000000000d9c000	0x0000000000d9d000	---	0x157000	8	1	1572910	/lib/libc-2.12.1.so
1980	0x0000000000d9d000	0x0000000000d9f000	r--	0x157000	8	1	1572910	/lib/libc-2.12.1.so
1980	0x0000000000d9f000	0x0000000000da0000	rw-	0x159000	8	1	1572910	/lib/libc-2.12.1.so
1980	0x0000000000da0000	0x0000000000da3000	rw-	0x0	0	0	0	
1980	**0x0000000008048000**	**0x0000000008049000**	**r-x**	**0x0**	**8**	**1**	**950507**	**/usr/_h4x_/_h4x_bd**
1980	0x0000000008049000	0x000000000804a000	r--	0x1000	8	1	950507	/usr/_h4x_/_h4x_bd
1980	0x000000000804a000	0x000000000804b000	rw-	0x2000	8	1	950507	/usr/_h4x_/_h4x_bd
1980	0x00000000b77d9000	0x00000000b77da000	rw-	0x0	0	0	0	
1980	0x00000000b77ec000	0x00000000b77ee000	rw-	0x0	0	0	0	
1980	0x00000000bffdc000	0x00000000bfffe000	rw-	0x0	0	0	0	[stack]

What this plugin reveals about this process is the actual location of the files associated with suspicious process *_h4x_bd*, specifically its actual location, */usr/_h4x_/_h4x_bd*. This process relies on two essential system libraries, the C library (*libc-X.so*) and the system linker/loader library (*ld-X.so*), where each represents some arbitrary version of said libraries. Even the inodes of these files are listed, thereby making it possible to dump the contents of these files to disk using a different Linux Volatility-based plugin.

The memory range of interest with respect to suspicious process 1980 has been highlighted in the above table. This memory range will likely contain the actual executable that represents process 1980 and it is revealed by its permission of *r-x*. Interestingly, this process only relies on two libraries, whereas most system processes rely on many additional libraries. For example, consider process 1550 (bash command shell), which relies on eight distinct library[8] files.

Finally, it is important to point out that the inode alluded to by this plugin is the actual disk-based inode, 950507, and is the very same as would be seen when performing command:

```
$       ls -ali /usr/_h4x_/_h4x_bd
```

[8] Under UNIX/Linux, a library file typically ends in *.so*.

The next plugin is used to identify additional information necessary to dump this process from the memory image.

3.5.3 Plugin linux_find_file

This particular plugin can be used to not only dump pre-identified files from the memory image (using information obtained from other plugins) but it can also list all filesystem objects with an open handle in memory. It will often list far more objects than *linux_proc_maps* or *linux_lsof*. However, it works differently from *linux_proc_maps* and *linux_lsof*. Thus, when seeking out abnormal libraries and process names, plugin *linux_lsof* should be used first, followed by *linux_proc_maps* then *linux_find_file* [10].

This particular plugin is used to identify the memory-specific inode for file */usr/_h4x_/_h4x_bd* that is required to dump it file from the memory image, assuming it still resides there (i.e., it has not been paged out).

The plugin was run using the following command:

```
$       volatility --profile=Linuxubuntu_10_10_profilex86 -f
        ubuntu_10_10_kbeast.mem linux_find_file -F "/usr/_h4x_/_h4x_bd"
```

This resulted in the following output:

```
Inode Number      Inode
--------------------  ---------------
   950507         0xf4db2cf8
```

Inode 0xf4db2cf8 is not actually an inode but the location in memory where this inode is stored. With this information, it should be possible to dump this process directly from the memory image using this same plugin. This is done using the following command:

```
$       volatility --profile=Linuxubuntu_10_10_profilex86 -f
        ubuntu_10_10_kbeast.mem linux_find_file -i 0xf4db2cf8  -O suspect.elf
```

This resulted in file *suspect.elf* being generated and dumped from the memory image. SHA1 analysis of the file reveals that it has the same hash as rootkit *_h4x_bd* as found in Section 3.1.3. Moreover, fuzzy hash matching indicated it was a 100% match.

Of course, in a real situation where only a memory image was available, an investigator/incident handler would not typically have the actual disk-based rootkit for comparative purposes.

3.5.4 Summary

Performing Volatility file detection and dumping has demonstrated that rootkits can sometimes be detected using alternate means. Under Ubuntu 10.10, where the rootkit was installed and loaded, the author had attempted to verify the location of the rootkit but both it and its directory were

hidden. However, using various Volatility memory plugins it was possible to not only identify it but also dump it. As it turned out, the dumped file *suspect.elf* is the actual rootkit, compiled and loaded into kernel space as per Section 1.6.

3.6 Step 6: Volatility kernel-specific analyses

This step will use one specific plugin to attempt to identify the presence of a kernel-level rootkit.

3.6.1 Plugin linux_lsmod

This plugin lists all visible Linux kernel modules running on the system, similar to the Linux *lsmod* command. Unlike *lsmod*, this plugin provides the base address for every detected kernel module.

The plugin was run using the following command:

```
$ volatility --profile=Linuxubuntu_10_10_profilex86 -f
ubuntu_10_10_kbeast.mem linux_lsmod -P
```

This resulted in the modules listed in Annex B.7. The *-P* parameter can be used to list all specified module input parameters. The *-S* parameter can be used to list all memory areas used by a given kernel module. Looking at the output, nothing appears out of the ordinary. All the kernel modules listed appear legitimate.

3.6.2 Summary

Performing Volatility kernel-specific analyses has demonstrated that listing kernel modules may not necessarily identify kernel-level rootkits.

Although the *linux_moddump* plugin could have been used to dump all kernel modules from the memory image the only way to reliably discern if any of them were infected would be to reverse engineer them all because it has been clearly demonstrated that none of the 51 scanners used by VirusTotal were capable of identifying the compiled rootkit.

3.7 Step 7: Volatility network-specific plugins

This step will use one specific plugin to attempt to identify the presence of a kernel-level rootkit.

3.7.1 Plugin linux_netstat

This plugin performs the equivalent of the UNIX/Linux *netstat* command that is used to print information concerning network connections (the actual *netstat* command does far more).

The plugin was run using the following command:

```
$       volatility --profile=Linuxubuntu_10_10_profilex86 -f
        ubuntu_10_10_kbeast.mem linux_netstat -v | grep -P '(TCP|UDP)'
```

The information revealed by the output shown below indicates that something suspicious is going on with respect to network communications.

```
UDP     0.0.0.0:5353  0.0.0.0:0         avahi-daemon/602
UDP     :::5353  :::0         avahi-daemon/602
UDP     0.0.0.0:48448 0.0.0.0:0         avahi-daemon/602
UDP     :::32796 :::0         avahi-daemon/602
UDP     0.0.0.0:68    0.0.0.0:0           dhclient/633
TCP     ::1:631   :::0    LISTEN          cupsd/653
TCP     127.0.0.1:631   0.0.0.0:0    LISTEN          cupsd/653
TCP     0.0.0.0:0     0.0.0.0:0    CLOSE        libvirtd/839
UDP     0.0.0.0:67    0.0.0.0:0           dnsmasq/982
TCP     192.168.122.1:53    0.0.0.0:0    LISTEN        dnsmasq/982
UDP     192.168.122.1:53    0.0.0.0:0           dnsmasq/982
TCP     0.0.0.0:13377 0.0.0.0:0    LISTEN        _h4x_bd/1980
```

It was found that process 1980 was listening on port 13377, in effect establishing a backdoor mechanism. All the other ports listed above should be considered normal.

3.7.2 Summary

Performing this specific Volatility network analysis has demonstrated that the rootkit has established a backdoor mechanism waiting for incoming connections.

However, in the author's opinion there is little reason to believe that other network-enabled plugins (e.g., *linux_arp* and *linux_route_cache*) would have revealed any additional information of interest.

3.8 Step 8: Volatility rootkit-specific analyses

3.8.1 Plugin linux_check_afinfo

This plugin validates two network protocol-specific kernel structures, *file_operations* and *sequence_operations* against kernel structures *tcp6_seq_afinfo*, *tcp4_seq_afinfo*,

udp6_seq_afinfo, *udp4_seq_afinfo*, *udplite6_seq_afinfo* and *udplite4_seq_afinfo*. Essentially, this plugin attempts to determine if any of these structures have been tampered with [10].

The plugin was run using the following command:

```
$       volatility --profile=Linuxubuntu_10_10_profilex86 -f
        ubuntu_10_10_kbeast.mem linux_check_afinfo
```

This resulted in the following output:

```
Symbol Name          Member       Address
-------------------- ------------ ----------------
tcp4_seq_afinfo      show         0xf96d0650
```

The above output indicates that the plugin has identified one network-specific structure, *tcp4_seq_afinfo*, which may have been modified by a rootkit.

3.8.2 Plugin linux_check_creds

This plugin is used to check for processes with raised privileges, typical of certain types of rootkits.

The plugin was run using the following command:

```
$       volatility --profile=Linuxubuntu_10_10_profilex86 -f
        ubuntu_10_10_kbeast.mem linux_check_fop
```

The plugin found no indication of process elevation.

3.8.3 Plugin linux_check_fop

An interesting plugin, it is used to verify if there are hooks in the kernel with respect to opened files and validates that each file's *file_operation* structure is intact. When a potential hook is discovered, the plugin will generate output [10].

The plugin was run using the following command:

```
$       volatility --profile=Linuxubuntu_10_10_profilex86 -f
        ubuntu_10_10_kbeast.mem linux_check_fop
```

Running this plugin produced no output, thereby indicating that no *file_operation* hook could be identified.

3.8.4 Plugin linux_check_idt

This plugin checks a memory image for signs of hooking in the system Interrupt Descriptor Table (IDT). If any of the IDTs appear to have been hooked, the plugin will issue HOOKED in lieu of the expected symbol name [10].

Issuing the following command resulted in the following output:

```
$ volatility --profile=Linuxubuntu_10_10_profilex86 -f
ubuntu_10_10_kbeast.mem linux_check_idt
```

The information in Table 8 indicates that nothing out of the ordinary has occurred to the system IDTs.

Table 8: Plugin output for linux_check_idt (sorted by index).

Index	Address	Symbol
0x0	0xc05c9708	divide_error
0x1	0xc05c97ac	debug
0x2	0xc05c9804	nmi
0x3	0xc05c9930	int3
0x4	0xc05c96b8	overflow
0x5	0xc05c96c4	bounds
0x6	0xc05c96d0	invalid_op
0x7	0xc05c96a4	device_not_available
0x9	0xc05c96dc	coprocessor_segment_overrun
0xa	0xc05c96e8	invalid_TSS
0xb	0xc05c96f0	segment_not_present
0xc	0xc05c96f8	stack_segment
0xd	0xc05c9968	general_protection
0xe	0xc05c972c	page_fault
0xf	0xc05c9720	spurious_interrupt_bug
0x10	0xc05c968c	coprocessor_error
0x11	0xc05c9700	alignment_check
0x12	0xc05c9714	machine_check
0x13	0xc05c9698	simd_coprocessor_error
0x80	0xc05c9058	system_call

3.8.5 Plugin linux_check_modules

Possibly the most powerful of Volatility's Linux-based rootkit checking plugins, the *linux_check_modules* plugin performs kernel module differencing. Specifically, it looks for inconsistencies between the kernel module list (found on running Linux systems using system command *lsmod* or kernel structure */proc/modules*) vs modules found under */sys/modules*.

The plugin was run using the following command:

```
$ volatility --profile=Linuxubuntu_10_10_profilex86 -f
ubuntu_10_10_kbeast.mem linux_check_modules
```

This resulted in the following important information being generated:

```
Module Name
-------------------
ipsecs_kbeast_v1
```

The plugin has positively identified a hidden kernel module with suspicious name KBeast, which is the name of a known open source Linux-based rootkit. However, other kernel-based plugins will be attempted in the hopes that they can perhaps shed more light on this rootkit.

3.8.6 Plugin linux_check_syscall

This plugin searches a memory image for hooked system calls (syscalls). If the plugin detects something, it will print HOOKED followed by the expected system call [10].

The plugin was run using the following command:

```
$ volatility --profile=Linuxubuntu_10_10_profilex86 -f
ubuntu_10_10_kbeast.mem linux_check_syscall
```

However, this plugin did not work under Fedora. No message concerning SLUB or SLAB was issued by the plugin. Instead, various messages were displayed indicating that there was an issue either with the plugin or with one or more missing Python dependencies.

3.8.7 Plugins linux_check_tty and linux_keyboard_notifier

Both plugins can be used to help identify kernel-level keyloggers as each plugin uses a different mechanism. It is hoped that one of them will determine if a keylogger is present in this memory image having been introduced by the rootkit.

The following commands were issued:

```
$ volatility --profile=Linuxubuntu_10_10_profilex86 -f
ubuntu_10_10_kbeast.mem linux_check_tty
```

```
$       volatility --profile=Linuxubuntu_10_10_profilex86 -f
        ubuntu_10_10_kbeast.mem linux_keyboard_notifier
```

However, both plugins were incapable of detecting the rootkit's keylogger. The following output was generated by the *linux_check_tty* (Table 9) whereas plugin *linux_keyboard_notifier* produced no output, respectively:

Table 9: Plugin output for linux_check_tty (sorted by tty).

Name	Address	Symbol
tty1	0xc03dd260	n_tty_receive_buf
tty2	0xc03dd260	n_tty_receive_buf
tty3	0xc03dd260	n_tty_receive_buf
tty4	0xc03dd260	n_tty_receive_buf
tty5	0xc03dd260	n_tty_receive_buf
tty6	0xc03dd260	n_tty_receive_buf
tty7	0xc03dd260	n_tty_receive_buf

Whereas the former plugin scans drivers for *tty* hooking, the latter plugin scans for hooked kernel callbacks [10].

3.8.8 Summary

Interestingly, this rootkit was advanced enough to evade most of Volatility's kernel-level rootkit identification plugins. The first of the only two plugins in this step that identified anything suspicious was *linux_check_afinfo,* which identified *tcp4_seq_afinfo* as suspicious. This is not particularly surprising as plugin *linux_netstat* identified a rogue TCP port.

The second plugin was *linux_check_modules* and it identified a hidden kernel module, KBeast, which is the name of a known Linux open source rootkit (see Section 3.8.5 for details). As it turns out, while rootkits under Linux have the ability to hide themselves from the */proc/modules* and *lsmod,* they cannot hide from */sys/modules* which keeps track of all kernel modules, hidden or not. According to [10], there are to date no Linux rootkits that are able to evade this detection.

Finally, even though the keylogger-checking plugins found nothing, it was ascertained that the keylogger did in fact work. The keylog dump file was found containing the various commands that had been typed into the console by the root user starting the moment the rootkit was loaded into kernel space.

4 Conclusion

It is difficult to compare KBeast to the capabilities of other Linux rootkits as they are less commonly encountered than those of Windows. In contrast, Windows rootkits are relatively common and there is a wide range in their capabilities: some are far less capable than KBeast while others are tremendously sophisticated both in terms of their ability to hide and in terms of their overall set of features. That said, based on the current capabilities of KBeast, it could be considered as slightly above average.

KBeast was not, however, exceptionally difficult to detect from the acquired memory image. Conversely, while running atop a live system under Ubuntu 10.10 it was much more difficult to identify; having tried through various system commands to list kernel modules as well as the rootkit's location on disk, it could not be found, as it was able to adequately hide itself. However, other commands including *lsof* and the killing of the rootkit's userland process, *_h4x_bd*, were not attempted.

While various Volatility plugins hinted at a suspicious process later identified as having both an open socket and network port, neither of which should have been in use, it was ultimately the module checking plugin, *linux_check_module*s, which identified the rootkit.

It was also rather surprising that at the time the rootkit sample was submitted to VirusTotal, not one of the 51 scanners detected it as either malicious or infected.

Finally, this case study will have hopefully demonstrated to others how to systematically proceed when investigating an infected Linux-based memory image.

References

[1] Ph03n1x. README.TXT. Rootkit documentation file. December 2011. http://ipsecs.org (Access date: 1 Aug 2014).

[2] Ph03n1x. Config.h. Source code configuration file. December 2011. http://ipsecs.org. (Access date: 1 Aug 2014).

[3] Volatility team. LinuxMemoryForensics: Instructions on how to access and use the Linux support. Online documentation. September 2013. http://code.google.com/p/volatility/wiki/LinuxMemoryForensics (Access date: 1 Aug 2014).

[4] Carbone, Richard and Bourdon-Richard, Sébastien. The definitive guide to Linux-based live memory acquisition tools: An addendum to "State of the art concerning memory acquisition software: A detailed examination of Linux, BSD and Solaris live memory acquisition". Technical Memorandum. TM 2012-319. DRDC – Valcartier Research Centre. September 2013.

[5] ForensicsWiki. Ssdeep. Online information. ForensicsWiki. October 2010. http://forensicswiki.org/wiki/Ssdeep (Access date: 1 Aug 2014).

[6] Carbone, Richard. Malware memory analysis for non-specialists: Investigating publicly available memory image for the Stuxnet worm. Scientific Report. SR DRDC-RDDC-2013-R1. DRDC – Valcartier Research Centre. November 2013.

[7] Carbone, Richard. Malware memory analysis for non-specialists: Investigating publicly available memory image for the Tigger Trojan horse. Scientific Report. SR DRDC-RDDC-2014-R28. DRDC – Valcartier Research Centre. June 2014.

[8] http://www.linuxvox.com/2012/06/what-is-the-kthreadd-process/ (Access date: 1 Aug 2014).

[9] YobiWiki. RAM analysis: Misc notes on physical RAM analysis. Online documentation. October 2013. http://wiki.yobi.be/wiki/RAM_analysis (Access date: 1 Aug 2014).

[10] Volatility. LinuxCommandReference23: A command reference for Linux. Online documentation. Unknown date. http://code.google.com/p/volatility/wiki/LinuxCommandReference23 (Access date: 1 Aug 2014).

[11] Kim, Joonsoo. How does the SLUB allocator work. Presentation. LGE CTO SWP Lab. Unknown date. http://events.linuxfoundation.org/images/stories/pdf/klf2012_kim.pdf (Access date: 1 Aug 2014).

[12] Wikipedia. Slab allocation. Online encyclopaedic entry. Wikimedia Foundation Inc. July 2014. http://en.wikipedia.org/wiki/Slab_allocation (Access date: 1 Aug 2014).

[13] Wikipedia. SLUB (software). Online encyclopaedic entry. Wikimedia Foundation Inc. August 2014. http://en.wikipedia.org/wiki/SLUB_(software) (Access date: 1 Aug 2014).

[14] Case, Andrew. Analyzing the KBeast Rootkit and Detecting Hidden Modules with Volatility. Blog. Infosec Island. September 2012. http://www.infosecisland.com/blogview/22410-Analyzing-the-KBeast-Rootkit-and-Detecting-Hidden-Modules-with-Volatility.html (Access date: 1 Aug 2014).

[15] Vandeven, Sally. Linux Rootkit Detection With OSSEC. SANS Gold paper. SANS. March 2014. http://www.sans.org/reading-room/whitepapers/detection/rootkit-detection-ossec-34555 (Access date: 1 Aug 2014).

[16] Xie, Xiongwei and Wang, Weicho. Rootkit Detection on Virtual Machines through Deep Information Extraction at Hypervisor-level. Technical paper. University of North Carolina, Charlotte. Presented in 4th International Workshop on Security and Privacy in Cloud Computing, 2013. 2013. http://webpages.uncc.edu/wwang22/Research/papers/Xie-SPCC-13.pdf (Access date: 1 Aug 2014).

Annex A Volatility Linux-based plugins

The following is a complete list of the default Linux-based plugins provided by Volatility 2.3.1:

Table A.1: List of Volatility 2.3.1 plugins.

Plugin	**Capability (as per Volatility --info output)**
linux_arp	Print the ARP table
linux_banner	Prints the Linux banner information
linux_bash	Recover bash history from bash process memory
linux_check_afinfo	Verifies the operation function pointers of network protocols
linux_check_creds	Checks if any processes are sharing credential structures
linux_check_evt_arm	Checks the Exception Vector Table to look for syscall table hooking
linux_check_fop	Check file operation structures for rootkit modifications
linux_check_idt	Checks if the IDT has been altered
linux_check_modules	Compares module list to sysfs info, if available
linux_check_syscall	Checks if the system call table has been altered
linux_check_syscall_arm	Checks if the system call table has been altered
linux_check_tty	Checks tty devices for hooks
linux_cpuinfo	Prints info about each active processor
linux_dentry_cache	Gather files from the dentry cache
linux_dmesg	Gather dmesg buffer
linux_dump_map	Writes selected memory mappings to disk
linux_find_file	Recovers tmpfs filesystems from memory
linux_ifconfig	Gathers active interfaces
linux_iomem	Provides output similar to /proc/iomem
linux_keyboard_notifier	Parses the keyboard notifier call chain
linux_lsmod	Gather loaded kernel modules
linux_lsof	Lists open files

Plugin	Capability (as per Volatility --info output)
linux_memmap	Dumps the memory map for linux tasks
linux_moddump	Extract loaded kernel modules
linux_mount	Gather mounted fs/devices
linux_mount_cache	Gather mounted fs/devices from kmem_cache
linux_netstat	Lists open sockets
linux_pidhashtable	Enumerates processes through the PID hash table
linux_pkt_queues	Writes per-process packet queues out to disk
linux_proc_maps	Gathers process maps for linux
linux_psaux	Gathers processes along with full command line and start time
linux_pslist	Gather active tasks by walking the task_struct->task list
linux_pslist_cache	Gather tasks from the kmem_cache
linux_pstree	Shows the parent/child relationship between processes
linux_psxview	Find hidden processes with various process listings
linux_route_cache	Recovers the routing cache from memory
linux_sk_buff_cache	Recovers packets from the sk_buff kmem_cache
linux_slabinfo	Mimics /proc/slabinfo on a running machine
linux_tmpfs	Recovers tmpfs filesystems from memory
linux_vma_cache	Gather VMAs from the vm_area_struct cache
linux_volshell	Shell in the memory image
linux_yarascan	A shell in the Linux memory image

Annex B Plugin output and listings

This annex provides the various output and listings for the different plugins used throughout this report which are too lengthy to fit within a given subsection.

B.1 Output for plugin linux_dmesg

The following output was generated by the Volatility *linux_dmesg* plugin, as found in Section 3.4.3:

```
[2314885531810281020.2314885531] ] Initializing cgroup subsys cpuset
<6>[    0.000000] Initializing cgroup subsys cpu
<5>[    0.000000] Linux version 2.6.35-22-generic (buildd@rothera) (gcc
version 4.4.5 (Ubuntu/Linaro 4.4.4-14ubuntu4) ) #33-Ubuntu SMP Sun Sep 19
20:34:50 UTC 2010 (Ubuntu 2.6.35-22.33-generic 2.6.35.4)
<6>[    0.000000] BIOS-provided physical RAM map:
<6>[        0.000000]    BIOS-e820:  0000000000000000  -  000000000009fc00
(usable)
<6>[        0.000000]    BIOS-e820:  000000000009fc00  -  00000000000a0000
(reserved)
<6>[        0.000000]    BIOS-e820:  00000000000f0000  -  0000000000100000
(reserved)
<6>[        0.000000]    BIOS-e820:  0000000000100000  -  000000007fff0000
(usable)
<6>[    0.000000]  BIOS-e820: 000000007fff0000 - 0000000080000000 (ACPI
data)
<6>[        0.000000]    BIOS-e820:  00000000fffc0000  -  0000000100000000
(reserved)
<5>[      0.000000] Notice: NX (Execute Disable) protection cannot be
enabled: non-PAE kernel!
<6>[    0.000000] DMI 2.5 present.
<7>[    0.000000] e820 update range: 0000000000000000 - 0000000000001000
(usable) ==> (reserved)
<7>[    0.000000] e820 remove range: 00000000000a0000 - 0000000000100000
(usable)
<6>[    0.000000] last_pfn = 0x7fff0 max_arch_pfn = 0x100000
<7>[    0.000000] MTRR default type: uncachable
<7>[    0.000000] MTRR variable ranges disabled:
<6>[      0.000000] x86 PAT enabled: cpu 0, old 0x7040600070406, new
0x7010600070106
<6>[    0.000000] CPU MTRRs all blank - virtualized system.
<7>[    0.000000] e820 update range: 0000000000002000 - 0000000000010000
(usable) ==> (reserved)
<6>[    0.000000] Scanning 1 areas for low memory corruption
<6>[    0.000000] modified physical RAM map:
<6>[        0.000000]    modified:  0000000000000000  -  0000000000001000
(reserved)
<6>[    0.000000]  modified: 0000000000001000 - 0000000000002000 (usable)
<6>[        0.000000]    modified:  0000000000002000  -  0000000000010000
(reserved)
<6>[    0.000000]  modified: 0000000000010000 - 000000000009fc00 (usable)
<6>[        0.000000]    modified:  000000000009fc00  -  00000000000a0000
(reserved)
<6>[        0.000000]    modified:  00000000000f0000  -  0000000000100000
(reserved)
<6>[    0.000000]  modified: 0000000000100000 - 000000007fff0000 (usable)
<6>[     0.000000]  modified: 000000007fff0000 - 0000000080000000 (ACPI
data)
<6>[        0.000000]    modified:  00000000fffc0000  -  0000000100000000
(reserved)
<7>[    0.000000] initial memory mapped : 0 - 00c00000
```

```
<6>[    0.000000] init_memory_mapping: 0000000000000000-00000000377fe000
<7>[    0.000000]  0000000000 - 0000400000 page 4k
<7>[    0.000000]  0000400000 - 0037400000 page 2M
<7>[    0.000000]  0037400000 - 00377fe000 page 4k
<7>[    0.000000] kernel direct mapping tables up to 377fe000 @ 15000-
1a000
<6>[    0.000000] RAMDISK: 375ad000 - 37ff0000
<6>[    0.000000] Allocated new RAMDISK: 009a5000 - 013e78c9
<6>[    0.000000] Move RAMDISK from 00000000375ad000 - 0000000037fef8c8
to 009a5000 - 013e78c8
<4>[    0.000000] ACPI: RSDP 000e0000 00024 (v02 VBOX  )
<4>[    0.000000] ACPI: XSDT 7fff0030 00034 (v01 VBOX   VBOXXSDT 00000001
ASL  00000061)
<4>[    0.000000] ACPI: FACP 7fff00f0 000F4 (v04 VBOX   VBOXFACP 00000001
ASL  00000061)
<4>[    0.000000] ACPI: DSDT 7fff0410 01B96 (v01 VBOX   VBOXBIOS 00000002
INTL 20100528)
<4>[    0.000000] ACPI: FACS 7fff0200 00040
<4>[    0.000000] ACPI: SSDT 7fff0240 001CC (v01 VBOX   VBOXCPUT 00000002
INTL 20100528)
<5>[    0.000000] 1159MB HIGHMEM available.
<5>[    0.000000] 887MB LOWMEM available.
<6>[    0.000000]   mapped low ram: 0 - 377fe000
<6>[    0.000000]   low ram: 0 - 377fe000
<4>[    0.000000] Zone PFN ranges:
<4>[    0.000000]   DMA      0x00000001 -> 0x00001000
<4>[    0.000000]   Normal   0x00001000 -> 0x000377fe
<4>[    0.000000]   HighMem  0x000377fe -> 0x0007fff0
<4>[    0.000000] Movable zone start PFN for each node
<4>[    0.000000] early_node_map[3] active PFN ranges
<4>[    0.000000]     0: 0x00000001 -> 0x00000002
<4>[    0.000000]     0: 0x00000010 -> 0x0000009f
<4>[    0.000000]     0: 0x00000100 -> 0x0007fff0
<7>[    0.000000] On node 0 totalpages: 524160
<7>[         0.000000]  free_area_init_node:  node  0,  pgdat  c07ffd40,
node_mem_map c13e9020
<7>[    0.000000]   DMA zone: 32 pages used for memmap
<7>[    0.000000]   DMA zone: 0 pages reserved
<7>[    0.000000]   DMA zone: 3952 pages, LIFO batch:0
<7>[    0.000000]   Normal zone: 1744 pages used for memmap
<7>[    0.000000]   Normal zone: 221486 pages, LIFO batch:31
<7>[    0.000000]   HighMem zone: 2320 pages used for memmap
<7>[    0.000000]   HighMem zone: 294626 pages, LIFO batch:31
<6>[    0.000000] Using APIC driver default
<6>[    0.000000] ACPI: PM-Timer IO Port: 0x4008
<6>[    0.000000] SMP: Allowing 1 CPUs, 0 hotplug CPUs
<6>[    0.000000] Found and enabled local APIC!
<7>[    0.000000] nr_irqs_gsi: 16
<6>[       0.000000]  PM:  Registered  nosave  memory:  0000000000002000  -
0000000000010000
<6>[       0.000000]  PM:  Registered  nosave  memory:  000000000009f000  -
00000000000a0000
<6>[       0.000000]  PM:  Registered  nosave  memory:  00000000000a0000  -
00000000000f0000
<6>[       0.000000]  PM:  Registered  nosave  memory:  00000000000f0000  -
0000000000100000
<6>[      0.000000] Allocating PCI resources starting at 80000000 (gap:
80000000:7ffc0000)
<6>[    0.000000] Booting paravirtualized kernel on bare hardware
<6>[     0.000000] setup_percpu: NR_CPUS:8 nr_cpumask_bits:8 nr_cpu_ids:1
nr_node_ids:1
<7>[    0.000000] early_res array is doubled to 64 at [16000 - 167ff]
<6>[      0.000000]  PERCPU:  Embedded  14  pages/cpu  @c2400000  s36416  r0
d20928 u4194304
<6>[    0.000000] pcpu-alloc: s36416 r0 d20928 u4194304 alloc=1*4194304
<6>[    0.000000] pcpu-alloc: [0] 0
<4>[    0.000000] Built 1 zonelists in Zone order, mobility grouping on.
Total pages: 520064
```

```
<5>[     0.000000] Kernel command line: BOOT_IMAGE=/boot/vmlinuz-2.6.35-
22-generic root=UUID=b13dedba-11eb-497f-96b2-e06d37b3aef1 ro quiet splash
<6>[    0.000000] PID hash table entries: 4096 (order: 2, 16384 bytes)
<6>[     0.000000] Dentry cache hash table entries: 131072 (order: 7,
524288 bytes)
<6>[    0.000000] Inode-cache hash table entries: 65536 (order: 6, 262144
bytes)
<6>[    0.000000] Enabling fast FPU save and restore... done.
<6>[    0.000000] Enabling unmasked SIMD FPU exception support... done.
<6>[    0.000000] Initializing CPU#0
<6>[    0.000000] allocated 10485420 bytes of page_cgroup
<6>[    0.000000] please try 'cgroup_disable=memory' option if you don't
want memory cgroups
<6>[    0.000000] Subtract (39 early reservations)
<6>[    0.000000]   #1 [0000001000 - 0000002000]   EX TRAMPOLINE
<6>[    0.000000]   #2 [0000100000 - 00009a0adc]   TEXT DATA BSS
<6>[    0.000000]   #3 [000009f800 - 0000100000]   BIOS reserved
<6>[    0.000000]   #4 [00009a1000 - 00009a410c]             BRK
<6>[    0.000000]   #5 [0000010000 - 0000011000]      TRAMPOLINE
<6>[    0.000000]   #6 [0000011000 - 0000015000]     ACPI WAKEUP
<6>[    0.000000]   #7 [0000015000 - 0000016000]         PGTABLE
<6>[    0.000000]   #8 [00009a5000 - 00013e8000]     NEW RAMDISK
<6>[    0.000000]   #9 [00013e8000 - 00013e9000]         BOOTMEM
<6>[    0.000000]   #10 [00013e9000 - 00023e9000]          BOOTMEM
<6>[    0.000000]   #11 [00023e9000 - 00023e9004]          BOOTMEM
<6>[    0.000000]   #12 [00023e9040 - 00023e9100]          BOOTMEM
<6>[    0.000000]   #13 [00023e9100 - 00023e9154]          BOOTMEM
<6>[    0.000000]   #14 [00023e9180 - 00023ec180]          BOOTMEM
<6>[    0.000000]   #15 [00023ec180 - 00023ec1f0]          BOOTMEM
<6>[    0.000000]   #16 [00023ec200 - 00023f2200]          BOOTMEM
<6>[    0.000000]   #17 [00023f2200 - 00023f22fc]          BOOTMEM
<6>[    0.000000]   #18 [00023f2300 - 00023f2340]          BOOTMEM
<6>[    0.000000]   #19 [00023f2340 - 00023f2380]          BOOTMEM
<6>[    0.000000]   #20 [00023f2380 - 00023f23c0]          BOOTMEM
<6>[    0.000000]   #21 [00023f23c0 - 00023f2400]          BOOTMEM
<6>[    0.000000]   #22 [00023f2400 - 00023f2440]          BOOTMEM
<6>[    0.000000]   #23 [00023f2440 - 00023f2480]          BOOTMEM
<6>[    0.000000]   #24 [00023f2480 - 00023f2490]          BOOTMEM
<6>[    0.000000]   #25 [00023f24c0 - 00023f24d0]          BOOTMEM
<6>[    0.000000]   #26 [00023f2500 - 00023f256a]          BOOTMEM
<6>[    0.000000]   #27 [00023f2580 - 00023f25ea]          BOOTMEM
<6>[    0.000000]   #28 [0002400000 - 000240e000]          BOOTMEM
<6>[    0.000000]   #29 [00023f4600 - 00023f4604]          BOOTMEM
<6>[    0.000000]   #30 [00023f4640 - 00023f4644]          BOOTMEM
<6>[    0.000000]   #31 [00023f4680 - 00023f4684]          BOOTMEM
<6>[    0.000000]   #32 [00023f46c0 - 00023f46c4]          BOOTMEM
<6>[    0.000000]   #33 [00023f4700 - 00023f47b0]          BOOTMEM
<6>[    0.000000]   #34 [00023f47c0 - 00023f4868]          BOOTMEM
<6>[    0.000000]   #35 [00023f4880 - 00023f8880]          BOOTMEM
<6>[    0.000000]   #36 [000240e000 - 000248e000]          BOOTMEM
<6>[    0.000000]   #37 [000248e000 - 00024ce000]          BOOTMEM
<6>[    0.000000]   #38 [00024ce000 - 0002ecdeac]          BOOTMEM
<6>[    0.000000] Initializing HighMem for node 0 (000377fe:0007fff0)
<6>[    0.000000] Memory: 2049740k/2097088k available (4928k kernel code,
46900k reserved, 2336k data, 684k init, 1187784k highmem)
<6>[    0.000000] virtual kernel memory layout:
<6>[    0.000000]     fixmap  : 0xfff16000 - 0xfffff000   ( 932 kB)
<6>[    0.000000]     pkmap   : 0xff800000 - 0xffc00000   (4096 kB)
<6>[    0.000000]     vmalloc : 0xf7ffe000 - 0xff7fe000   ( 120 MB)
<6>[    0.000000]     lowmem  : 0xc0000000 - 0xf77fe000   ( 887 MB)
<6>[    0.000000]       .init : 0xc0819000 - 0xc08c4000   ( 684 kB)
<6>[    0.000000]       .data : 0xc05d029e - 0xc0818668   (2336 kB)
<6>[    0.000000]       .text : 0xc0100000 - 0xc05d029e   (4928 kB)
<6>[     0.000000] Checking if this processor honours the WP bit even in
supervisor mode...Ok.
<6>[    0.000000] SLUB: Genslabs=13, HWalign=64, Order=0-3, MinObjects=0,
CPUs=1, Nodes=1
<6>[    0.000000] Hierarchical RCU implementation.
```

```
<6>[    0.000000]   RCU   dyntick-idle   grace-period   acceleration   is
enabled.
<6>[    0.000000]   RCU-based detection of stalled CPUs is disabled.
<6>[    0.000000]   Verbose stalled-CPUs detection is disabled.
<6>[    0.000000] NR_IRQS:2304 nr_irqs:256
<4>[    0.000000] Console: colour VGA+ 80x25
<6>[    0.000000] console [tty0] enabled
<4>[    0.000000] Fast TSC calibration failed
<4>[    0.000000] TSC: Unable to calibrate against PIT
<6>[    0.000000] TSC: using PMTIMER reference calibration
<4>[    0.000000] Detected 2633.297 MHz processor.
<6>[      0.016027] Calibrating delay loop (skipped), value calculated
using timer frequency.. 5266.59 BogoMIPS (lpj=10533188)
<6>[    0.016036] pid_max: default: 32768 minimum: 301
<6>[    0.016094] Security Framework initialized
<6>[    0.016156] AppArmor: AppArmor initialized
<6>[    0.016158] Yama: becoming mindful.
<4>[    0.016291] Mount-cache hash table entries: 512
<6>[    0.016586] Initializing cgroup subsys ns
<6>[    0.016599] Initializing cgroup subsys cpuacct
<6>[    0.016603] Initializing cgroup subsys memory
<6>[    0.016616] Initializing cgroup subsys devices
<6>[    0.016618] Initializing cgroup subsys freezer
<6>[    0.016630] Initializing cgroup subsys net_cls
<6>[    0.016765] mce: CPU supports 0 MCE banks
<6>[    0.016810] using mwait in idle threads.
<6>[    0.016826] Performance Events: unsupported p6 CPU model 15 no PMU
driver, software events only.
<6>[    0.025156] SMP alternatives: switching to UP code
<6>[    0.271961] Freeing SMP alternatives: 24k freed
<6>[    0.272806] ACPI: Core revision 20100428
<4>[    0.275287] ACPI: setting ELCR to 0200 (from 0e00)
<6>[    0.275457] ftrace: converting mcount calls to 0f 1f 44 00 00
<6>[    0.275600] ftrace: allocating 21758 entries in 43 pages
<6>[    0.278849] Enabling APIC mode:  Flat.  Using 0 I/O APICs
<4>[    0.280682] weird, boot CPU (#0) not listed by the BIOS.
<5>[    0.280871] SMP motherboard not detected.
<6>[    0.284000] SMP disabled
<6>[    0.284000] Brought up 1 CPUs
<6>[    0.284000] Total of 1 processors activated (5266.59 BogoMIPS).
<6>[    0.284000] devtmpfs: initialized
<6>[    0.284000] regulator: core version 0.5
<4>[    0.284000] Time: 17:41:21  Date: 05/23/14
<6>[    0.284000] NET: Registered protocol family 16
<6>[    0.284000] EISA bus registered
<6>[    0.284000] ACPI: bus type pci registered
<6>[      0.284000] PCI: PCI BIOS revision 2.10 entry at 0xfda26, last
bus=0
<6>[    0.284000] PCI: Using configuration type 1 fo
[2332073165573077042.2332073165] bled
<6>[    0.284000] Brought up 1 CPUs
<6>[    0.284000] Total of 1 processors activated (5266.59 BogoMIPS).
<6>[    0.284000] devtmpfs: initialized
<6>[    0.284000] regulator: core version 0.5
<4>[    0.284000] Time: 17:41:21  Date: 05/23/14
<6>[    0.284000] NET: Registered protocol family 16
<6>[    0.284000] EISA bus registered
<6>[    0.284000] ACPI: bus type pci registered
<6>[      0.284000] PCI: PCI BIOS revision 2.10 entry at 0xfda26, last
bus=0
<6>[    0.284000] PCI: Using configuration type 1 for base access
<4>[    0.284022] bio: create slab <bio-0> at 0
<7>[    0.284524] ACPI: EC: Look up EC in DSDT
<4>[    0.284978] ACPI: Executed 1 blocks of module-level executable AML
code
<6>[    0.288198] ACPI: Interpreter enabled
<6>[    0.288202] ACPI: (supports S0 S5)
<6>[    0.288213] ACPI: Using PIC for interrupt routing
```

```
<6>[    0.290462] ACPI: No dock devices found.
<6>[       0.290467] PCI: Ignoring host bridge windows from ACPI; if
necessary, use "pci=use_crs" and report a bug
<6>[    0.290535] ACPI: PCI Root Bridge [PCI0] (domain 0000 [bus 00-ff])
<7>[     0.290668] pci_root PNP0A03:00: host bridge window [io  0x0000-
0x0cf7] (ignored)
<7>[     0.290671] pci_root PNP0A03:00: host bridge window [io  0x0d00-
0xffff] (ignored)
<7>[        0.290673] pci_root PNP0A03:00: host bridge window [mem
0x000a0000-0x000bffff] (ignored)
<7>[        0.290675] pci_root PNP0A03:00: host bridge window [mem
0x80000000-0xffdfffff] (ignored)
<7>[    0.292302] pci 0000:00:01.1: reg 20: [io  0xd000-0xd00f]
<7>[     0.300637] pci 0000:00:02.0: reg 10: [mem 0xe0000000-0xe7ffffff
pref]
<7>[    0.304979] pci 0000:00:03.0: reg 10: [mem 0xf0000000-0xf001ffff]
<7>[    0.305349] pci 0000:00:03.0: reg 18: [io  0xd010-0xd017]
<7>[    0.306456] pci 0000:00:04.0: reg 10: [io  0xd020-0xd03f]
<7>[    0.309303] pci 0000:00:04.0: reg 14: [mem 0xf0400000-0xf07fffff]
<7>[     0.313410] pci 0000:00:04.0: reg 18: [mem 0xf0800000-0xf0803fff
pref]
<7>[    0.314393] pci 0000:00:05.0: reg 10: [io  0xd100-0xd1ff]
<7>[    0.314768] pci 0000:00:05.0: reg 14: [io  0xd200-0xd23f]
<7>[    0.320885] pci 0000:00:06.0: reg 10: [mem 0xf0804000-0xf0804fff]
<7>[    0.325323] pci 0000:00:0b.0: reg 10: [mem 0xf0805000-0xf0805fff]
<7>[    0.326565] pci 0000:00:0d.0: reg 10: [io  0xd240-0xd247]
<7>[    0.326972] pci 0000:00:0d.0: reg 18: [io  0xd250-0xd257]
<7>[    0.328748] pci 0000:00:0d.0: reg 20: [io  0xd260-0xd26f]
<7>[    0.332784] pci 0000:00:0d.0: reg 24: [mem 0xf0806000-0xf0807fff]
<7>[    0.333624] pci_bus 0000:00: on NUMA node 0
<7>[    0.333865] ACPI: PCI Interrupt Routing Table [\_SB_.PCI0._PRT]
<6>[    0.342005] ACPI: PCI Interrupt Link [LNKA] (IRQs 5 9 10 *11)
<6>[    0.342332] ACPI: PCI Interrupt Link [LNKB] (IRQs 5 9 10 *11)
<6>[    0.342443] ACPI: PCI Interrupt Link [LNKC] (IRQs 5 9 *10 11)
<6>[    0.342553] ACPI: PCI Interrupt Link [LNKD] (IRQs 5 *9 10 11)
<6>[    0.342627] HEST: Table is not found!
<6>[                        0.342915]      vgaarb:      device      added:
PCI:0000:00:02.0,decodes=io+mem,owns=io+mem,locks=none
<6>[    0.342920] vgaarb: loaded
<5>[    0.343350] SCSI subsystem initialized
<7>[    0.343565] libata version 3.00 loaded.
<6>[    0.343674] usbcore: registered new interface driver usbfs
<6>[    0.343695] usbcore: registered new interface driver hub
<6>[    0.343784] usbcore: registered new device driver usb
<6>[    0.344000] ACPI: WMI: Mapper loaded
<6>[    0.344000] PCI: Using ACPI for IRQ routing
<7>[    0.344000] PCI: pci_cache_line_size set to 64 bytes
<7>[    0.344293] reserve RAM buffer: 0000000000002000 - 000000000000ffff
<7>[    0.344300] reserve RAM buffer: 000000000009fc00 - 000000000009ffff
<7>[    0.344302] reserve RAM buffer: 000000007fff0000 - 000000007fffffff
<6>[    0.344590] NetLabel: Initializing
<6>[    0.344596] NetLabel:  domain hash size = 128
<6>[    0.344597] NetLabel:  protocols = UNLABELED CIPSOv4
<6>[    0.344738] NetLabel:  unlabeled traffic allowed by default
<6>[    0.344804] Switching to clocksource tsc
<6>[    0.357854] AppArmor: AppArmor Filesystem Enabled
<6>[    0.357904] pnp: PnP ACPI init
<6>[    0.357942] ACPI: bus type pnp registered
<3>[    0.358132] ERROR: Unable to locate IOAPIC for GSI 1
<3>[    0.358274] ERROR: Unable to locate IOAPIC for GSI 12
<3>[    0.358332] ERROR: Unable to locate IOAPIC for GSI 7
<6>[    0.359137] pnp: PnP ACPI: found 5 devices
<6>[    0.359139] ACPI: ACPI bus type pnp unregistered
<6>[    0.359148] PnPBIOS: Disabled by ACPI PNP
<7>[    0.395354] pci_bus 0000:00: resource 0 [io  0x0000-0xffff]
<7>[    0.395356] pci_bus 0000:00: resource 1 [mem 0x00000000-0xffffffff]
<6>[    0.395426] NET: Registered protocol family 2
```

```
<6>[     0.395518] IP route cache hash table entries: 32768 (order: 5,
131072 bytes)
<6>[     0.395995] TCP established hash table entries: 131072 (order: 8,
1048576 bytes)
<6>[     0.399103] TCP bind hash table entries: 65536 (order: 7, 524288
bytes)
<6>[     0.399103] TCP: Hash tables configured (established 131072 bind
65536)
<6>[    0.399103] TCP reno registered
<6>[    0.399103] UDP hash table entries: 512 (order: 2, 16384 bytes)
<6>[      0.399103] UDP-Lite hash table entries: 512 (order: 2, 16384
bytes)
<6>[    0.399103] NET: Registered protocol family 1
<6>[    0.399103] pci 0000:00:00.0: Limiting direct PCI/PCI transfers
<6>[    0.400078] pci 0000:00:01.0: Activating ISA DMA hang workarounds
<7>[    0.400125] pci 0000:00:02.0: Boot video device
<7>[    0.400407] PCI: CLS 0 bytes, default 64
<6>[     0.400572] platform rtc_cmos: registered platform RTC device (no
PNP device found)
<6>[    0.400712] cpufreq-nforce2: No nForce2 chipset.
<6>[    0.400739] Scanning for low memory corruption every 60 seconds
<6>[    0.400948] audit: initializing netlink socket (disabled)
<5>[    0.400963] type=2000 audit(1400866881.400:1): initialized
<6>[    0.410250] Trying to unpack rootfs image as initramfs...
<4>[    0.428089] highmem bounce pool size: 64 pages
<6>[      0.428151] HugeTLB registered 4 MB page size, pre-allocated 0
pages
<5>[    0.436453] VFS: Disk quotas dquot_6.5.2
<4>[      0.436511] Dquot-cache hash table entries: 1024 (order 0, 4096
bytes)
<6>[    0.436992] fuse init (API version 7.14)
<6>[    0.437090] msgmni has been set to 1683
<6>[      0.451742] Block layer SCSI generic (bsg) driver version 0.4
loaded (major 253)
<6>[    0.451792] io scheduler noop registered
<6>[    0.451838] io scheduler deadline registered
<6>[    0.451892] io scheduler cfq registered (default)
<6>[    0.452021] pci_hotplug: PCI Hot Plug PCI Core version: 0.5
<6>[    0.452081] pciehp: PCI Express Hot Plug Controller Driver version:
0.4
<6>[    0.452410] ACPI: AC Adapter [AC] (on-line)
<6>[                    0.452503]     input:     Power     Button     as
/devices/LNXSYSTM:00/LNXPWRBN:00/input/input0
<6>[    0.452543] ACPI: Power Button [PWRF]
<6>[                    0.452651]     input:     Sleep     Button     as
/devices/LNXSYSTM:00/LNXSLPBN:00/input/input1
<6>[    0.452681] ACPI: Sleep Button [SLPF]
<7>[    0.452969] ACPI: acpi_idle registered with cpuidle
<6>[    0.454057] ERST: Table is not found!
<6>[    0.454420] Serial: 8250/16550 driver, 4 ports, IRQ sharing enabled
<6>[    0.455564] brd: module loaded
<6>[    0.455863] loop: module loaded
<7>[    0.456049] ata_piix 0000:00:01.1: version 2.13
<7>[    0.456150] ata_piix 0000:00:01.1: setting latency timer to 64
<6>[    0.456290] isapnp: Scanning for PnP cards...
<6>[    0.461943] scsi0 : ata_piix
<6>[    0.462026] scsi1 : ata_piix
<6>[    0.462061] ata1: PATA max UDMA/33 cmd 0x1f0 ctl 0x3f6 bmdma 0xd000
irq 14
<6>[    0.462068] ata2: PATA max UDMA/33 cmd 0x170 ctl 0x376 bmdma 0xd008
irq 15
<6>[    0.464017] Fixed MDIO Bus: probed
<6>[    0.464101] PPP generic driver version 2.4.2
<6>[    0.464184] tun: Universal TUN/TAP device driver, 1.6
<6>[    0.464219] tun: (C) 1999-2004 Max Krasnyansky <maxk@qualcomm.com>
<6>[      0.464350] ehci_hcd: USB 2.0 'Enhanced' Host Controller (EHCI)
Driver
<4>[    0.464950] ACPI: PCI Interrupt Link [LNKC] enabled at IRQ 10
```

```
<7>[    0.464981] PCI: setting IRQ 10 as level-triggered
<6>[    0.465019] ehci_hcd 0000:00:0b.0: PCI INT A -> Link[LNKC] -> GSI
10 (level, low) -> IRQ 10
<7>[    0.465125] ehci_hcd 0000:00:0b.0: setting latency timer to 64
<6>[    0.465176] ehci_hcd 0000:00:0b.0: EHCI Host Controller
<6>[    0.465256] ehci_hcd 0000:00:0b.0: new USB bus registered, assigned
bus number 1
<6
[7301496515958096672.7301496515]      56.526213]  lo:  Disabled  Privacy
Extensions
```

B.2 Output for plugin linux_psaux

The following output was generated by the Volatility *linux_psaux* plugin, as found in Section 3.3.1:

Table B.1: Plugin output for linux_psaux (sorted by PID).

PID	UID	GID	Arguments
1	0	0	/sbin/init ro quiet splash
2	0	0	[kthreadd]
3	0	0	[ksoftirqd/0]
4	0	0	[migration/0]
5	0	0	[watchdog/0]
6	0	0	[events/0]
7	0	0	[cpuset]
8	0	0	[khelper]
9	0	0	[netns]
10	0	0	[async/mgr]
11	0	0	[pm]
12	0	0	[sync_supers]
13	0	0	[bdi-default]
14	0	0	[kintegrityd/0]
15	0	0	[kblockd/0]
16	0	0	[kacpid]
17	0	0	[kacpi_notify]
18	0	0	[kacpi_hotplug]
19	0	0	[ata_aux]
20	0	0	[ata_sff/0]
21	0	0	[khubd]
22	0	0	[kseriod]
23	0	0	[kmmcd]
25	0	0	[khungtaskd]
26	0	0	[kswapd0]
27	0	0	[ksmd]
28	0	0	[aio/0]
29	0	0	[ecryptfs-kthrea]
30	0	0	[crypto/0]
36	0	0	[scsi_eh_0]
37	0	0	[scsi_eh_1]
40	0	0	[kstriped]

PID	UID	GID	Arguments
41	0	0	[kmpathd/0]
42	0	0	[kmpath_handlerd]
43	0	0	[ksnapd]
44	0	0	[kondemand/0]
45	0	0	[kconservative/0]
155	0	0	[scsi_eh_2]
164	0	0	[usbhid_resumer]
180	0	0	[jbd2/sda1-8]
181	0	0	[ext4-dio-unwrit]
222	0	0	[flush-8:0]
248	0	0	upstart-udev-bridge --daemon
251	0	0	udevd --daemon
378	0	0	udevd --daemon
379	0	0	[iprt/0]
383	0	0	udevd --daemon
403	0	0	[kpsmoused]
570	101	103	rsyslogd -c4
581	102	105	dbus-daemon --system --fork
599	0	0	NetworkManager
602	104	109	avahi-daemon: ru
605	104	109	avahi-daemon: ch
610	0	0	/usr/sbin/modem-manager
630	0	0	/sbin/wpa_supplicant -u -s
633	0	0	/sbin/dhclient -d -sf /usr/lib/NetworkManager/nm-dhcp-client.action -pf /var/run/dhclient-eth0.pid -lf /var/lib/dhcp3/dhclient-6579e68b-f570-4b95-9640-eca69940d467-eth0.lease -cf /var/run/nm-dhclient-eth0.conf eth0
636	0	0	gdm-binary
653	0	0	/usr/sbin/cupsd -C /etc/cups/cupsd.conf
657	0	0	/usr/sbin/console-kit-daemon --no-daemon
725	0	0	/usr/lib/gdm/gdm-simple-slave --display-id /org/gnome/DisplayManager/Display1
745	0	0	/usr/bin/X :0 -nr -verbose -auth /var/run/gdm/auth-for-gdm-myRUZd/database -nolisten tcp vt7
801	0	0	/sbin/getty -8 38400 tty4
805	0	0	/sbin/getty -8 38400 tty5
813	0	0	/sbin/getty -8 38400 tty2

PID	UID	GID	Arguments
815	0	0	/sbin/getty -8 38400 tty3
817	0	0	/sbin/getty -8 38400 tty6
822	0	0	anacron -s
823	0	0	acpid -c /etc/acpi/events -s /var/run/acpid.socket
826	0	0	cron
827	0	0	atd
839	0	0	/usr/sbin/libvirtd -d
939	0	0	/usr/sbin/VBoxService
965	0	1000	/usr/lib/gdm/gdm-session-worker
982	65534	30	dnsmasq --strict-order --bind-interfaces --pid-file=/var/run/libvirt/network/default.pid --conf-file= --listen-address 192.168.122.1 --except-interface lo --dhcp-range 192.168.122.2,192.168.122.254 --dhcp-lease-max=253
999	1000	1000	gnome-session
1099	1000	1000	/usr/bin/VBoxClient --clipboard
1115	1000	1000	/usr/bin/VBoxClient --display
1128	1000	1000	/usr/bin/VBoxClient --seamless
1140	1000	1000	/usr/bin/VBoxClient --draganddrop
1146	1000	1000	/usr/bin/ssh-agent /usr/bin/dbus-launch --exit-with-session gnome-session
1151	1000	1000	/usr/bin/dbus-launch --exit-with-session gnome-session
1153	1000	1000	/bin/dbus-daemon --fork --print-pid 5 --print-address 7 --session
1182	1000	1000	/usr/lib/libgconf2-4/gconfd-2
1201	1000	1000	gnome-power-manager
1205	0	0	/sbin/getty -8 38400 tty1
1210	1000	1000	/usr/bin/gnome-keyring-daemon --start --components=ssh
1221	0	0	/usr/lib/upower/upowerd
1226	1000	1000	/usr/lib/gnome-settings-daemon/gnome-settings-daemon
1235	1000	1000	/usr/lib/gvfs/gvfsd
1255	1000	1000	bluetooth-applet
1256	1000	1000	/usr/lib/gvfs//gvfs-fuse-daemon /home/richard/.gvfs
1259	1000	1000	/usr/bin/compiz
1261	1000	1000	/usr/lib/evolution/2.30/evolution-alarm-notify
1266	0	0	/usr/lib/policykit-1/polkitd
1267	1000	1000	/usr/bin/pulseaudio --start --log-target=syslog
1271	1000	1000	nautilus

PID	UID	GID	Arguments
1273	110	117	/usr/lib/rtkit/rtkit-daemon
1275	1000	1000	gnome-panel
1278	1000	1000	nm-applet --sm-disable
1287	1000	1000	/usr/lib/policykit-1-gnome/polkit-gnome-authentication-agent-1
1310	1000	1000	/usr/lib/pulseaudio/pulse/gconf-helper
1350	1000	1000	/usr/lib/gvfs/gvfsd-trash --spawner :1.13 /org/gtk/gvfs/exec_spaw/0
1368	1000	1000	/usr/lib/gvfs/gvfs-gdu-volume-monitor
1376	1000	1000	/usr/lib/bonobo-activation/bonobo-activation-server --ac-activate --ior-output-fd=20
1391	0	0	/usr/lib/udisks/udisks-daemon
1395	1000	1000	gnome-screensaver
1406	0	0	udisks-daemon: polling /dev/sr
1421	1000	1000	/usr/lib/gnome-panel/wnck-applet --oaf-activate-iid=OAFIID:GNOME_Wncklet_Factory --oaf-ior-fd=22
1424	1000	1000	/usr/lib/gnome-applets/trashapplet --oaf-activate-iid=OAFIID:GNOME_Panel_TrashApplet_Factory --oaf-ior-fd=30
1431	1000	1000	/usr/lib/gvfs/gvfs-gphoto2-volume-monitor
1445	1000	1000	/usr/lib/gvfs/gvfs-afc-volume-monitor
1452	1000	1000	/usr/lib/gnome-disk-utility/gdu-notification-daemon
1453	1000	1000	/usr/lib/gvfs/gvfsd-burn --spawner :1.13 /org/gtk/gvfs/exec_spaw/1
1463	1000	1000	/usr/lib/indicator-applet/indicator-applet-session --oaf-activate-iid=OAFIID:GNOME_FastUserSwitchApplet_Factory --oaf-ior-fd=25
1465	1000	1000	/usr/lib/gnome-panel/clock-applet --oaf-activate-iid=OAFIID:GNOME_ClockApplet_Factory --oaf-ior-fd=36
1466	1000	1000	/usr/lib/gnome-panel/notification-area-applet --oaf-activate-iid=OAFIID:GNOME_NotificationAreaApplet_Factory --oaf-ior-fd=42
1469	1000	1000	/usr/lib/indicator-applet/indicator-applet --oaf-activate-iid=OAFIID:GNOME_IndicatorApplet_Factory --oaf-ior-fd=43
1485	1000	1000	/usr/lib/gvfs/gvfsd-metadata
1487	1000	1000	/usr/lib/indicator-sound/indicator-sound-service
1489	1000	1000	/usr/lib/indicator-messages/indicator-messages-service
1490	1000	1000	/bin/sh -c /usr/bin/compiz-decorator

PID	UID	GID	Arguments
1492	1000	1000	/usr/lib/indicator-application/indicator-application-service
1493	1000	1000	/usr/bin/gtk-window-decorator
1500	1000	1000	/usr/lib/indicator-session/indicator-session-service
1502	1000	1000	/usr/lib/indicator-me/indicator-me-service
1514	1000	1000	gnome-terminal
1517	1000	1000	/usr/bin/python /usr/share/system-config-printer/applet.py
1519	1000	1000	gnome-pty-helper
1521	1000	1000	bash
1542	0	0	su - root
1550	0	0	-su
1611	1000	1000	update-notifier
1614	0	0	[flush-vboxsf-1]
1980	**2**	**2**	**./_h4x_bd**

B.3 Output for plugin linux_pslist

The following output was generated by the Volatility *linux_pslist* plugin, as found in Section 3.3.2:

Table B.2: Plugin output for linux_pslist (sorted by PID).

Offset	Name	PID	UID	GID	DTB	Start Time
0xf7070000	init	1	0	0	0x368ea000	2014-05-23 17:41:21 UTC+0000
0xf7070cb0	kthreadd	2	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf7071960	ksoftirqd/0	3	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf7072610	migration/0	4	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf70732c0	watchdog/0	5	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf7073f70	events/0	6	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf7074c20	cpuset	7	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf70758d0	khelper	8	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf7076580	netns	9	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf7077230	async/mgr	10	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf7098000	pm	11	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf7098cb0	sync_supers	12	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf7099960	bdi-default	13	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf709a610	kintegrityd/0	14	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf709b2c0	kblockd/0	15	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf709bf70	kacpid	16	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf709cc20	kacpi_notify	17	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf709d8d0	kacpi_hotplug	18	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf709e580	ata_aux	19	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf709f230	ata_sff/0	20	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf70f8000	khubd	21	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf70f8cb0	kseriod	22	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf70f9960	kmmcd	23	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf70fb2c0	khungtaskd	25	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf70fbf70	kswapd0	26	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf70fcc20	ksmd	27	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf70fd8d0	aio/0	28	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf70fe580	ecryptfs-kthrea	29	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf70ff230	crypto/0	30	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf73f8000	scsi_eh_0	36	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf73fa610	scsi_eh_1	37	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf73fbf70	kstriped	40	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf73fcc20	kmpathd/0	41	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf73fd8d0	kmpath_handlerd	42	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf73fe580	ksnapd	43	0	0	----------	2014-05-23 17:41:21 UTC+0000

Offset	Name	PID	UID	GID	DTB	Start Time		
0xf73ff230	kondemand/0	44	0	0	----------	2014-05-23	17:41:21	UTC+0000
0xf6878000	kconservative/0	45	0	0	----------	2014-05-23	17:41:21	UTC+0000
0xf696b2c0	scsi_eh_2	155	0	0	----------	2014-05-23	17:41:24	UTC+0000
0xf696a610	usbhid_resumer	164	0	0	----------	2014-05-23	17:41:24	UTC+0000
0xf6968cb0	jbd2/sda1-8	180	0	0	----------	2014-05-23	17:41:25	UTC+0000
0xf6969960	ext4-dio-unwrit	181	0	0	----------	2014-05-23	17:41:25	UTC+0000
0xf6908000	flush-8:0	222	0	0	----------	2014-05-23	17:41:26	UTC+0000
0xf692b2c0	upstart-udev-br	248	0	0	0x36904000	2014-05-23	17:41:26	UTC+0000
0xf692a610	udevd	251	0	0	0x36a90000	2014-05-23	17:41:27	UTC+0000
0xf59032c0	udevd	378	0	0	0x358a5000	2014-05-23	17:41:29	UTC+0000
0xf5906580	iprt/0	379	0	0	----------	2014-05-23	17:41:29	UTC+0000
0xf5903f70	udevd	383	0	0	0x358af000	2014-05-23	17:41:29	UTC+0000
0xf58858d0	kpsmoused	403	0	0	----------	2014-05-23	17:41:30	UTC+0000
0xf5858cb0	rsyslogd	570	101	103	0x35957000	2014-05-23	17:41:34	UTC+0000
0xf5880cb0	dbus-daemon	581	102	105	0x358d6000	2014-05-23	17:41:35	UTC+0000
0xf58e1960	NetworkManager	599	0	0	0x36981000	2014-05-23	17:41:35	UTC+0000
0xf58e2610	avahi-daemon	602	104	109	0x3583d000	2014-05-23	17:41:35	UTC+0000
0xf58e0000	avahi-daemon	605	104	109	0x36b99000	2014-05-23	17:41:35	UTC+0000
0xf58e7230	modem-manager	610	0	0	0x35952000	2014-05-23	17:41:35	UTC+0000
0xf5907230	wpa_supplicant	630	0	0	0x3590b000	2014-05-23	17:41:36	UTC+0000
0xf687f230	dhclient	633	0	0	0x358d3000	2014-05-23	17:41:36	UTC+0000
0xf6b78cb0	gdm-binary	636	0	0	0x35898000	2014-05-23	17:41:36	UTC+0000
0xf58e0cb0	cupsd	653	0	0	0x358ce000	2014-05-23	17:41:37	UTC+0000
0xf6bd8000	console-kit-dae	657	0	0	0x3583c000	2014-05-23	17:41:37	UTC+0000
0xf5b132c0	gdm-simple-slav	725	0	0	0x35912000	2014-05-23	17:41:37	UTC+0000
0xf58e32c0	Xorg	745	0	0	0x35915000	2014-05-23	17:41:38	UTC+0000
0xf5be1960	getty	801	0	0	0x36ba2000	2014-05-23	17:41:40	UTC+0000
0xf5be7230	getty	805	0	0	0x36978000	2014-05-23	17:41:40	UTC+0000
0xf6bdcc20	getty	813	0	0	0x358ac000	2014-05-23	17:41:40	UTC+0000
0xf6bda610	getty	815	0	0	0x36bb8000	2014-05-23	17:41:40	UTC+0000
0xf690a610	getty	817	0	0	0x36bb7000	2014-05-23	17:41:40	UTC+0000
0xf6908cb0	anacron	822	0	0	0x36b4d000	2014-05-23	17:41:41	UTC+0000
0xf690bf70	acpid	823	0	0	0x36b49000	2014-05-23	17:41:41	UTC+0000
0xf6928cb0	cron	826	0	0	0x36b5a000	2014-05-23	17:41:41	UTC+0000
0xf692d8d0	atd	827	0	0	0x36bf9000	2014-05-23	17:41:41	UTC+0000
0xf6b7bf70	libvirtd	839	0	0	0x35840000	2014-05-23	17:41:41	UTC+0000
0xf5900000	VBoxService	939	0	0	0x35959000	2014-05-23	17:41:44	UTC+0000
0xf585bf70	gdm-session-wor	965	0	1000	0x36bc3000	2014-05-23	17:41:45	UTC+0000
0xf6bde580	dnsmasq	982	65534	30	0x358a9000	2014-05-23	17:41:45	UTC+0000
0xf690f230	gnome-session	999	1000	1000	0x35056000	2014-05-23	17:41:46	UTC+0000

Offset	Name	PID	UID	GID	DTB	Start Time		
0xf6bdd8d0	VBoxClient	1099	1000	1000	0x36b61000	2014-05-23	17:41:48	UTC+0000
0xf6909960	VBoxClient	1115	1000	1000	0x35baa000	2014-05-23	17:41:49	UTC+0000
0xf6bdbf70	VBoxClient	1128	1000	1000	0x35bcf000	2014-05-23	17:41:49	UTC+0000
0xf5818000	VBoxClient	1140	1000	1000	0x35bda000	2014-05-23	17:41:49	UTC+0000
0xf585cc20	ssh-agent	1146	1000	1000	0x36bf2000	2014-05-23	17:41:49	UTC+0000
0xf692bf70	dbus-launch	1151	1000	1000	0x35bae000	2014-05-23	17:41:49	UTC+0000
0xf5068cb0	dbus-daemon	1153	1000	1000	0x3504d000	2014-05-23	17:41:49	UTC+0000
0xf506e580	gconfd-2	1182	1000	1000	0x35059000	2014-05-23	17:41:50	UTC+0000
0xf5be0cb0	gnome-power-man	1201	1000	1000	0x36b40000	2014-05-23	17:41:51	UTC+0000
0xf6b79960	getty	1205	0	0	0x35079000	2014-05-23	17:41:51	UTC+0000
0xf5be32c0	gnome-keyring-d	1210	1000	1000	0x35085000	2014-05-23	17:41:51	UTC+0000
0xf506f230	upowerd	1221	0	0	0x358c6000	2014-05-23	17:41:52	UTC+0000
0xf50b2610	gnome-settings-	1226	1000	1000	0x3508c000	2014-05-23	17:41:52	UTC+0000
0xf50b3f70	gvfsd	1235	1000	1000	0x3508e000	2014-05-23	17:41:53	UTC+0000
0xf58e6580	bluetooth-apple	1255	1000	1000	0x35081000	2014-05-23	17:41:53	UTC+0000
0xf6bdf230	gvfs-fuse-daemo	1256	1000	1000	0x350d2000	2014-05-23	17:41:53	UTC+0000
0xf50b4c20	compiz	1259	1000	1000	0x351a3000	2014-05-23	17:41:53	UTC+0000
0xf50b32c0	evolution-alarm	1261	1000	1000	0x350e3000	2014-05-23	17:41:54	UTC+0000
0xf506d8d0	polkitd	1266	0	0	0x3511e000	2014-05-23	17:41:54	UTC+0000
0xf5108000	pulseaudio	1267	1000	1000	0x350f2000	2014-05-23	17:41:54	UTC+0000
0xf510a610	nautilus	1271	1000	1000	0x3511d000	2014-05-23	17:41:54	UTC+0000
0xf510bf70	rtkit-daemon	1273	110	117	0x3512a000	2014-05-23	17:41:54	UTC+0000
0xf5109960	gnome-panel	1275	1000	1000	0x35126000	2014-05-23	17:41:54	UTC+0000
0xf510e580	nm-applet	1278	1000	1000	0x35131000	2014-05-23	17:41:54	UTC+0000
0xf5168000	polkit-gnome-au	1287	1000	1000	0x35149000	2014-05-23	17:41:55	UTC+0000
0xf50b0cb0	gconf-helper	1310	1000	1000	0x3517d000	2014-05-23	17:41:56	UTC+0000
0xf516b2c0	gvfsd-trash	1350	1000	1000	0x351e0000	2014-05-23	17:41:59	UTC+0000
0xf5b17230	gvfs-gdu-volume	1368	1000	1000	0x350bf000	2014-05-23	17:42:00	UTC+0000
0xf516cc20	bonobo-activati	1376	1000	1000	0x35210000	2014-05-23	17:42:00	UTC+0000
0xf690cc20	udisks-daemon	1391	0	0	0x35202000	2014-05-23	17:42:01	UTC+0000
0xf516d8d0	gnome-screensav	1395	1000	1000	0x351e9000	2014-05-23	17:42:01	UTC+0000
0xf5237230	udisks-daemon	1406	0	0	0x351f4000	2014-05-23	17:42:01	UTC+0000
0xf5233f70	wnck-applet	1421	1000	1000	0x35249000	2014-05-23	17:42:02	UTC+0000
0xf52332c0	trashapplet	1424	1000	1000	0x35245000	2014-05-23	17:42:02	UTC+0000
0xf5274c20	gvfs-gphoto2-vo	1431	1000	1000	0x351af000	2014-05-23	17:42:02	UTC+0000
0xf5273f70	gvfs-afc-volume	1445	1000	1000	0x350e0000	2014-05-23	17:42:03	UTC+0000
0xf5271960	gdu-notificatio	1452	1000	1000	0x35258000	2014-05-23	17:42:03	UTC+0000
0xf5270cb0	gvfsd-burn	1453	1000	1000	0x351dd000	2014-05-23	17:42:04	UTC+0000
0xf52358d0	indicator-apple	1463	1000	1000	0x352a6000	2014-05-23	17:42:04	UTC+0000
0xf5234c20	clock-applet	1465	1000	1000	0x3523a000	2014-05-23	17:42:04	UTC+0000

Offset	Name	PID	UID	GID	DTB	Start Time		
0xf50b58d0	notification-ar	1466	1000	1000	0x35268000	2014-05-23	17:42:04	UTC+0000
0xf516e580	indicator-apple	1469	1000	1000	0x351f5000	2014-05-23	17:42:04	UTC+0000
0xf510f230	gvfsd-metadata	1485	1000	1000	0x3529d000	2014-05-23	17:42:07	UTC+0000
0xf5068000	indicator-sound	1487	1000	1000	0x352a1000	2014-05-23	17:42:07	UTC+0000
0xf5236580	indicator-messa	1489	1000	1000	0x352e2000	2014-05-23	17:42:08	UTC+0000
0xf51cd8d0	sh	1490	1000	1000	0x352f9000	2014-05-23	17:42:08	UTC+0000
0xf5276580	indicator-appli	1492	1000	1000	0x352fa000	2014-05-23	17:42:08	UTC+0000
0xf52758d0	gtk-window-deco	1493	1000	1000	0x35309000	2014-05-23	17:42:08	UTC+0000
0xf51cf230	indicator-sessi	1500	1000	1000	0x35329000	2014-05-23	17:42:10	UTC+0000
0xf5348000	indicator-me-se	1502	1000	1000	0x35336000	2014-05-23	17:42:10	UTC+0000
0xf534b2c0	gnome-terminal	1514	1000	1000	0x3535e000	2014-05-23	17:42:23	UTC+0000
0xf534d8d0	applet.py	1517	1000	1000	0x3536b000	2014-05-23	17:42:23	UTC+0000
0xf534bf70	gnome-pty-helpe	1519	1000	1000	0x35365000	2014-05-23	17:42:24	UTC+0000
0xf534e580	bash	1521	1000	1000	0x35384000	2014-05-23	17:42:24	UTC+0000
0xf5858000	su	1542	0	0	0x3580b000	2014-05-23	17:42:29	UTC+0000
0xf506b2c0	bash	1550	0	0	0x35bea000	2014-05-23	17:42:32	UTC+0000
0xf6928000	update-notifier	1611	1000	1000	0x35076000	2014-05-23	17:42:54	UTC+0000
0xf6b7e580	flush-vboxsf-1	1614	0	0	---------	2014-05-23	17:42:56	UTC+0000
0xf690b2c0	**_h4x_bd**	**1980**	**2**	**2**	**0x353f6000**	**2014-05-23**	**17:44:58**	**UTC+0000**

B.4 Output for plugin linux_pstree

The following output was generated by the Volatility *linux_pslist* plugin, as found in Section 3.3.4:

Table B.3: Plugin output for linux_pstree.

Name	Pid	Uid
init	1	0
.upstart-udev-br	248	0
.udevd	251	0
..udevd	378	0
..udevd	383	0
.rsyslogd	570	101
.dbus-daemon	581	102
.NetworkManager	599	0
..dhclient	633	0
.avahi-daemon	602	104
..avahi-daemon	605	104
.modem-manager	610	0
.gdm-binary	636	0
..gdm-simple-slav	725	0
...Xorg	745	0
...gdm-session-wor	965	0
....gnome-session	999	1000
.....ssh-agent	1146	1000
.....gnome-power-man	1201	1000
.....bluetooth-apple	1255	1000
.....compiz	1259	1000
......sh	1490	1000
.......gtk-window-deco	1493	1000
.....evolution-alarm	1261	1000
.....nautilus	1271	1000
.....gnome-panel	1275	1000
.....nm-applet	1278	1000
.....polkit-gnome-au	1287	1000
.....gdu-notificatio	1452	1000
.....applet.py	1517	1000
.....update-notifier	1611	1000
.wpa_supplicant	630	0
.cupsd	653	0

Name	Pid	Uid
.console-kit-dae	657	0
.getty	801	0
.getty	805	0
.getty	813	0
.getty	815	0
.getty	817	0
.anacron	822	0
.acpid	823	0
.cron	826	0
.atd	827	0
.libvirtd	839	0
.VBoxService	939	0
.dnsmasq	982	65534
.VBoxClient	1099	1000
.VBoxClient	1115	1000
.VBoxClient	1128	1000
.VBoxClient	1140	1000
.dbus-daemon	1153	1000
.dbus-launch	1151	1000
.gconfd-2	1182	1000
.getty	1205	0
.gnome-keyring-d	1210	1000
.upowerd	1221	0
.gvfsd	1235	1000
.gnome-settings-	1226	1000
.gvfs-fuse-daemo	1256	1000
.rtkit-daemon	1273	110
.polkitd	1266	0
.pulseaudio	1267	1000
..gconf-helper	1310	1000
.gvfsd-trash	1350	1000
.bonobo-activati	1376	1000
.gnome-screensav	1395	1000
.udisks-daemon	1391	0
..udisks-daemon	1406	0
.trashapplet	1424	1000
.wnck-applet	1421	1000
.gvfs-gdu-volume	1368	1000
.gvfs-gphoto2-vo	1431	1000
.gvfs-afc-volume	1445	1000

Name	Pid	Uid
.gvfsd-burn	1453	1000
.indicator-apple	1463	1000
.clock-applet	1465	1000
.indicator-apple	1469	1000
.notification-ar	1466	1000
.gvfsd-metadata	1485	1000
.indicator-sound	1487	1000
.indicator-appli	1492	1000
.indicator-messa	1489	1000
.indicator-sessi	1500	1000
.indicator-me-se	1502	1000
.gnome-terminal	1514	1000
..gnome-pty-helpe	1519	1000
..bash	1521	1000
...su	1542	0
....bash	1550	0
._h4x_bd	1980	2
[kthreadd]	2	0
.[ksoftirqd/0]	3	0
.[migration/0]	4	0
.[watchdog/0]	5	0
.[events/0]	6	0
.[cpuset]	7	0
.[khelper]	8	0
.[netns]	9	0
.[async/mgr]	10	0
.[pm]	11	0
.[sync_supers]	12	0
.[bdi-default]	13	0
.[kintegrityd/0]	14	0
.[kblockd/0]	15	0
.[kacpid]	16	0
.[kacpi_notify]	17	0
.[kacpi_hotplug]	18	0
.[ata_aux]	19	0
.[ata_sff/0]	20	0
.[khubd]	21	0
.[kseriod]	22	0
.[kmmcd]	23	0
.[khungtaskd]	25	0

Name	Pid	Uid
.[kswapd0]	26	0
.[ksmd]	27	0
.[aio/0]	28	0
.[ecryptfs-kthrea]	29	0
.[crypto/0]	30	0
.[scsi_eh_0]	36	0
.[scsi_eh_1]	37	0
.[kstriped]	40	0
.[kmpathd/0]	41	0
.[kmpath_handlerd]	42	0
.[ksnapd]	43	0
.[kondemand/0]	44	0
.[kconservative/0]	45	0
.[scsi_eh_2]	155	0
.[usbhid_resumer]	164	0
.[jbd2/sda1-8]	180	0
.[ext4-dio-unwrit]	181	0
.[flush-8:0]	222	0
.[iprt/0]	379	0
.[kpsmoused]	403	0
.[flush-vboxsf-1]	1614	0

B.5 Output for plugin linux_pidhashtable

The following output was generated by the Volatility *linux_pidhashtable* plugin, as found in Section 3.3.5:

Table B.4: Plugin output for linux_pidhashtable (sorted by PID).

Offset	Name	Pid	Uid	Gid	DTB	Start Time
0xf7070000	init	1	0	0	0x368ea000	2014-05-23 17:41:21 UTC+0000
0xf7070cb0	kthreadd	2	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf7071960	ksoftirqd/0	3	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf7072610	migration/0	4	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf70732c0	watchdog/0	5	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf7073f70	events/0	6	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf7074c20	cpuset	7	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf70758d0	khelper	8	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf7076580	netns	9	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf7077230	async/mgr	10	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf7098000	pm	11	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf7098cb0	sync_supers	12	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf7099960	bdi-default	13	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf709a610	kintegrityd/0	14	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf709b2c0	kblockd/0	15	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf709bf70	kacpid	16	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf709cc20	kacpi_notify	17	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf709d8d0	kacpi_hotplug	18	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf709e580	ata_aux	19	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf709f230	ata_sff/0	20	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf70f8000	khubd	21	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf70f8cb0	kseriod	22	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf70f9960	kmmcd	23	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf70fb2c0	khungtaskd	25	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf70fbf70	kswapd0	26	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf70fcc20	ksmd	27	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf70fd8d0	aio/0	28	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf70fe580	ecryptfs-kthrea	29	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf70ff230	crypto/0	30	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf73f8000	scsi_eh_0	36	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf73fa610	scsi_eh_1	37	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf73fbf70	kstriped	40	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf73fcc20	kmpathd/0	41	0	0	----------	2014-05-23 17:41:21 UTC+0000

Offset	Name	Pid	Uid	Gid	DTB	Start Time
0xf73fd8d0	kmpath_handlerd	42	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf73fe580	ksnapd	43	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf73ff230	kondemand/0	44	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf6878000	kconservative/0	45	0	0	----------	2014-05-23 17:41:21 UTC+0000
0xf696b2c0	scsi_eh_2	155	0	0	----------	2014-05-23 17:41:24 UTC+0000
0xf696a610	usbhid_resumer	164	0	0	----------	2014-05-23 17:41:24 UTC+0000
0xf6968cb0	jbd2/sda1-8	180	0	0	----------	2014-05-23 17:41:25 UTC+0000
0xf6969960	ext4-dio-unwrit	181	0	0	----------	2014-05-23 17:41:25 UTC+0000
0xf6908000	flush-8:0	222	0	0	----------	2014-05-23 17:41:26 UTC+0000
0xf692b2c0	upstart-udev-br	248	0	0	0x36904000	2014-05-23 17:41:26 UTC+0000
0xf692a610	udevd	251	0	0	0x36a90000	2014-05-23 17:41:27 UTC+0000
0xf59032c0	udevd	378	0	0	0x358a5000	2014-05-23 17:41:29 UTC+0000
0xf5906580	iprt/0	379	0	0	----------	2014-05-23 17:41:29 UTC+0000
0xf5903f70	udevd	383	0	0	0x358af000	2014-05-23 17:41:29 UTC+0000
0xf58858d0	kpsmoused	403	0	0	----------	2014-05-23 17:41:30 UTC+0000
0xf5858cb0	rsyslogd	570	101	103	0x35957000	2014-05-23 17:41:34 UTC+0000
0xf690e580	rsyslogd	575	101	103	0x35957000	2014-05-23 17:41:35 UTC+0000
0xf5883f70	rsyslogd	576	101	103	0x35957000	2014-05-23 17:41:35 UTC+0000
0xf5880cb0	dbus-daemon	581	102	105	0x358d6000	2014-05-23 17:41:35 UTC+0000
0xf58e1960	NetworkManager	599	0	0	0x36981000	2014-05-23 17:41:35 UTC+0000
0xf58e2610	avahi-daemon	602	104	109	0x3583d000	2014-05-23 17:41:35 UTC+0000
0xf58e0000	avahi-daemon	605	104	109	0x36b99000	2014-05-23 17:41:35 UTC+0000
0xf58e7230	modem-manager	610	0	0	0x35952000	2014-05-23 17:41:35 UTC+0000
0xf5907230	wpa_supplicant	630	0	0	0x3590b000	2014-05-23 17:41:36 UTC+0000
0xf687f230	dhclient	633	0	0	0x358d3000	2014-05-23 17:41:36 UTC+0000
0xf6b78cb0	gdm-binary	636	0	0	0x35898000	2014-05-23 17:41:36 UTC+0000
0xf585e580	NetworkManager	642	0	0	0x36981000	2014-05-23 17:41:36 UTC+0000
0xf58e0cb0	cupsd	653	0	0	0x358ce000	2014-05-23 17:41:37 UTC+0000
0xf6bd8000	console-kit-dae	657	0	0	0x3583c000	2014-05-23 17:41:37 UTC+0000
0xf59bb2c0	console-kit-dae	658	0	0	0x3583c000	2014-05-23 17:41:37 UTC+0000
0xf59b8000	console-kit-dae	659	0	0	0x3583c000	2014-05-23 17:41:37 UTC+0000
0xf59bf230	console-kit-dae	660	0	0	0x3583c000	2014-05-23 17:41:37 UTC+0000
0xf59bd8d0	console-kit-dae	661	0	0	0x3583c000	2014-05-23 17:41:37 UTC+0000
0xf59b8cb0	console-kit-dae	662	0	0	0x3583c000	2014-05-23 17:41:37 UTC+0000
0xf59b9960	console-kit-dae	663	0	0	0x3583c000	2014-05-23 17:41:37 UTC+0000
0xf59be580	console-kit-dae	664	0	0	0x3583c000	2014-05-23 17:41:37 UTC+0000
0xf6968000	console-kit-dae	665	0	0	0x3583c000	2014-05-23 17:41:37 UTC+0000
0xf5900cb0	console-kit-dae	666	0	0	0x3583c000	2014-05-23 17:41:37 UTC+0000
0xf59058d0	console-kit-dae	667	0	0	0x3583c000	2014-05-23 17:41:37 UTC+0000
0xf5904c20	console-kit-dae	668	0	0	0x3583c000	2014-05-23 17:41:37 UTC+0000

Offset	Name	Pid	Uid	Gid	DTB	Start Time
0xf5902610	console-kit-dae	669	0	0	0x3583c000	2014-05-23 17:41:37 UTC+0000
0xf5901960	console-kit-dae	670	0	0	0x3583c000	2014-05-23 17:41:37 UTC+0000
0xf581b2c0	console-kit-dae	671	0	0	0x3583c000	2014-05-23 17:41:37 UTC+0000
0xf581a610	console-kit-dae	672	0	0	0x3583c000	2014-05-23 17:41:37 UTC+0000
0xf6b7f230	console-kit-dae	673	0	0	0x3583c000	2014-05-23 17:41:37 UTC+0000
0xf6b7a610	console-kit-dae	674	0	0	0x3583c000	2014-05-23 17:41:37 UTC+0000
0xf6b7cc20	console-kit-dae	675	0	0	0x3583c000	2014-05-23 17:41:37 UTC+0000
0xf585a610	console-kit-dae	676	0	0	0x3583c000	2014-05-23 17:41:37 UTC+0000
0xf585b2c0	console-kit-dae	677	0	0	0x3583c000	2014-05-23 17:41:37 UTC+0000
0xf5859960	console-kit-dae	678	0	0	0x3583c000	2014-05-23 17:41:37 UTC+0000
0xf6929960	console-kit-dae	679	0	0	0x3583c000	2014-05-23 17:41:37 UTC+0000
0xf692e580	console-kit-dae	680	0	0	0x3583c000	2014-05-23 17:41:37 UTC+0000
0xf692cc20	console-kit-dae	681	0	0	0x3583c000	2014-05-23 17:41:37 UTC+0000
0xf58832c0	console-kit-dae	682	0	0	0x3583c000	2014-05-23 17:41:37 UTC+0000
0xf5882610	console-kit-dae	683	0	0	0x3583c000	2014-05-23 17:41:37 UTC+0000
0xf5880000	console-kit-dae	684	0	0	0x3583c000	2014-05-23 17:41:37 UTC+0000
0xf58e3f70	console-kit-dae	685	0	0	0x3583c000	2014-05-23 17:41:37 UTC+0000
0xf5ab8000	console-kit-dae	686	0	0	0x3583c000	2014-05-23 17:41:37 UTC+0000
0xf5ab8cb0	console-kit-dae	687	0	0	0x3583c000	2014-05-23 17:41:37 UTC+0000
0xf5ab9960	console-kit-dae	688	0	0	0x3583c000	2014-05-23 17:41:37 UTC+0000
0xf5aba610	console-kit-dae	689	0	0	0x3583c000	2014-05-23 17:41:37 UTC+0000
0xf5abb2c0	console-kit-dae	690	0	0	0x3583c000	2014-05-23 17:41:37 UTC+0000
0xf5abbf70	console-kit-dae	691	0	0	0x3583c000	2014-05-23 17:41:37 UTC+0000
0xf5abcc20	console-kit-dae	692	0	0	0x3583c000	2014-05-23 17:41:37 UTC+0000
0xf5abd8d0	console-kit-dae	693	0	0	0x3583c000	2014-05-23 17:41:37 UTC+0000
0xf5abe580	console-kit-dae	694	0	0	0x3583c000	2014-05-23 17:41:37 UTC+0000
0xf5abf230	console-kit-dae	695	0	0	0x3583c000	2014-05-23 17:41:37 UTC+0000
0xf5ad8000	console-kit-dae	696	0	0	0x3583c000	2014-05-23 17:41:37 UTC+0000
0xf5ad8cb0	console-kit-dae	697	0	0	0x3583c000	2014-05-23 17:41:37 UTC+0000
0xf5ad9960	console-kit-dae	698	0	0	0x3583c000	2014-05-23 17:41:37 UTC+0000
0xf5ada610	console-kit-dae	699	0	0	0x3583c000	2014-05-23 17:41:37 UTC+0000
0xf5adb2c0	console-kit-dae	700	0	0	0x3583c000	2014-05-23 17:41:37 UTC+0000
0xf5adbf70	console-kit-dae	701	0	0	0x3583c000	2014-05-23 17:41:37 UTC+0000
0xf5adcc20	console-kit-dae	702	0	0	0x3583c000	2014-05-23 17:41:37 UTC+0000
0xf5add8d0	console-kit-dae	703	0	0	0x3583c000	2014-05-23 17:41:37 UTC+0000
0xf5ade580	console-kit-dae	704	0	0	0x3583c000	2014-05-23 17:41:37 UTC+0000
0xf5adf230	console-kit-dae	705	0	0	0x3583c000	2014-05-23 17:41:37 UTC+0000
0xf5af0000	console-kit-dae	706	0	0	0x3583c000	2014-05-23 17:41:37 UTC+0000
0xf5af0cb0	console-kit-dae	707	0	0	0x3583c000	2014-05-23 17:41:37 UTC+0000
0xf5af1960	console-kit-dae	708	0	0	0x3583c000	2014-05-23 17:41:37 UTC+0000

Offset	Name	Pid	Uid	Gid	DTB	Start Time
0xf5af2610	console-kit-dae	709	0	0	0x3583c000	2014-05-23 17:41:37 UTC+0000
0xf5af32c0	console-kit-dae	710	0	0	0x3583c000	2014-05-23 17:41:37 UTC+0000
0xf5af3f70	console-kit-dae	711	0	0	0x3583c000	2014-05-23 17:41:37 UTC+0000
0xf5af4c20	console-kit-dae	712	0	0	0x3583c000	2014-05-23 17:41:37 UTC+0000
0xf5af58d0	console-kit-dae	713	0	0	0x3583c000	2014-05-23 17:41:37 UTC+0000
0xf5af6580	console-kit-dae	714	0	0	0x3583c000	2014-05-23 17:41:37 UTC+0000
0xf5af7230	console-kit-dae	715	0	0	0x3583c000	2014-05-23 17:41:37 UTC+0000
0xf5b10000	console-kit-dae	716	0	0	0x3583c000	2014-05-23 17:41:37 UTC+0000
0xf5b10cb0	console-kit-dae	717	0	0	0x3583c000	2014-05-23 17:41:37 UTC+0000
0xf5b11960	console-kit-dae	718	0	0	0x3583c000	2014-05-23 17:41:37 UTC+0000
0xf5b12610	console-kit-dae	719	0	0	0x3583c000	2014-05-23 17:41:37 UTC+0000
0xf5b14c20	console-kit-dae	722	0	0	0x3583c000	2014-05-23 17:41:37 UTC+0000
0xf5b132c0	gdm-simple-slav	725	0	0	0x35912000	2014-05-23 17:41:37 UTC+0000
0xf5b158d0	gdm-binary	726	0	0	0x35898000	2014-05-23 17:41:38 UTC+0000
0xf58e32c0	Xorg	745	0	0	0x35915000	2014-05-23 17:41:38 UTC+0000
0xf581f230	gdm-simple-slav	747	0	0	0x35912000	2014-05-23 17:41:38 UTC+0000
0xf5be1960	getty	801	0	0	0x36ba2000	2014-05-23 17:41:40 UTC+0000
0xf5be7230	getty	805	0	0	0x36978000	2014-05-23 17:41:40 UTC+0000
0xf6bdcc20	getty	813	0	0	0x358ac000	2014-05-23 17:41:40 UTC+0000
0xf6bda610	getty	815	0	0	0x36bb8000	2014-05-23 17:41:40 UTC+0000
0xf690a610	getty	817	0	0	0x36bb7000	2014-05-23 17:41:40 UTC+0000
0xf6908cb0	anacron	822	0	0	0x36b4d000	2014-05-23 17:41:41 UTC+0000
0xf690bf70	acpid	823	0	0	0x36b49000	2014-05-23 17:41:41 UTC+0000
0xf6928cb0	cron	826	0	0	0x36b5a000	2014-05-23 17:41:41 UTC+0000
0xf692d8d0	atd	827	0	0	0x36bf9000	2014-05-23 17:41:41 UTC+0000
0xf6b7bf70	libvirtd	839	0	0	0x35840000	2014-05-23 17:41:41 UTC+0000
0xf5819960	libvirtd	845	0	0	0x35840000	2014-05-23 17:41:41 UTC+0000
0xf581cc20	libvirtd	846	0	0	0x35840000	2014-05-23 17:41:41 UTC+0000
0xf581d8d0	libvirtd	847	0	0	0x35840000	2014-05-23 17:41:41 UTC+0000
0xf581bf70	libvirtd	848	0	0	0x35840000	2014-05-23 17:41:41 UTC+0000
0xf5818cb0	libvirtd	849	0	0	0x35840000	2014-05-23 17:41:41 UTC+0000
0xf5884c20	libvirtd	850	0	0	0x35840000	2014-05-23 17:41:41 UTC+0000
0xf5900000	VBoxService	939	0	0	0x35959000	2014-05-23 17:41:44 UTC+0000
0xf5887230	control	941	0	0	0x35959000	2014-05-23 17:41:44 UTC+0000
0xf5b13f70	timesync	942	0	0	0x35959000	2014-05-23 17:41:44 UTC+0000
0xf5b16580	vminfo	943	0	0	0x35959000	2014-05-23 17:41:44 UTC+0000
0xf5be6580	cpuhotplug	944	0	0	0x35959000	2014-05-23 17:41:44 UTC+0000
0xf5be2610	memballoon	945	0	0	0x35959000	2014-05-23 17:41:44 UTC+0000
0xf58e4c20	vmstats	946	0	0	0x35959000	2014-05-23 17:41:44 UTC+0000
0xf6bdb2c0	automount	947	0	0	0x35959000	2014-05-23 17:41:44 UTC+0000

Offset	Name	Pid	Uid	Gid	DTB	Start Time
0xf585bf70	gdm-session-wor	965	0	1000	0x36bc3000	2014-05-23 17:41:45 UTC+0000
0xf6bde580	dnsmasq	982	65534	30	0x358a9000	2014-05-23 17:41:45 UTC+0000
0xf690f230	gnome-session	999	1000	1000	0x35056000	2014-05-23 17:41:46 UTC+0000
0xf690d8d0	gdm-session-wor	1000	0	1000	0x36bc3000	2014-05-23 17:41:46 UTC+0000
0xf6bdd8d0	VBoxClient	1099	1000	1000	0x36b61000	2014-05-23 17:41:48 UTC+0000
0xf696bf70	SHCLIP	1103	1000	1000	0x36b61000	2014-05-23 17:41:48 UTC+0000
0xf6909960	VBoxClient	1115	1000	1000	0x35baa000	2014-05-23 17:41:49 UTC+0000
0xf585d8d0	X11 monitor	1121	1000	1000	0x35baa000	2014-05-23 17:41:49 UTC+0000
0xf6bdbf70	VBoxClient	1128	1000	1000	0x35bcf000	2014-05-23 17:41:49 UTC+0000
0xf692f230	Host events	1131	1000	1000	0x35bcf000	2014-05-23 17:41:49 UTC+0000
0xf5818000	VBoxClient	1140	1000	1000	0x35bda000	2014-05-23 17:41:49 UTC+0000
0xf6bd9960	HGCM-NOTIFY	1143	1000	1000	0x35bda000	2014-05-23 17:41:49 UTC+0000
0xf6bd8cb0	X11-NOTIFY	1144	1000	1000	0x35bda000	2014-05-23 17:41:49 UTC+0000
0xf585cc20	ssh-agent	1146	1000	1000	0x36bf2000	2014-05-23 17:41:49 UTC+0000
0xf692bf70	dbus-launch	1151	1000	1000	0x35bae000	2014-05-23 17:41:49 UTC+0000
0xf5068cb0	dbus-daemon	1153	1000	1000	0x3504d000	2014-05-23 17:41:49 UTC+0000
0xf506bf70	gnome-session	1178	1000	1000	0x35056000	2014-05-23 17:41:50 UTC+0000
0xf506e580	gconfd-2	1182	1000	1000	0x35059000	2014-05-23 17:41:50 UTC+0000
0xf5be0cb0	gnome-power-man	1201	1000	1000	0x36b40000	2014-05-23 17:41:51 UTC+0000
0xf506a610	gnome-session	1203	1000	1000	0x35056000	2014-05-23 17:41:51 UTC+0000
0xf6b79960	getty	1205	0	0	0x35079000	2014-05-23 17:41:51 UTC+0000
0xf5be32c0	gnome-keyring-d	1210	1000	1000	0x35085000	2014-05-23 17:41:51 UTC+0000
0xf5be58d0	gnome-keyring-d	1211	1000	1000	0x35085000	2014-05-23 17:41:51 UTC+0000
0xf5be0000	gnome-keyring-d	1212	1000	1000	0x35085000	2014-05-23 17:41:52 UTC+0000
0xf506cc20	gnome-power-man	1216	1000	1000	0x36b40000	2014-05-23 17:41:52 UTC+0000
0xf506f230	upowerd	1221	0	0	0x358c6000	2014-05-23 17:41:52 UTC+0000
0xf50b2610	gnome-settings-	1226	1000	1000	0x3508c000	2014-05-23 17:41:52 UTC+0000
0xf50b1960	gnome-settings-	1228	1000	1000	0x3508c000	2014-05-23 17:41:52 UTC+0000
0xf50b3f70	gvfsd	1235	1000	1000	0x3508e000	2014-05-23 17:41:53 UTC+0000
0xf58e6580	bluetooth-apple	1255	1000	1000	0x35081000	2014-05-23 17:41:53 UTC+0000
0xf6bdf230	gvfs-fuse-daemo	1256	1000	1000	0x350d2000	2014-05-23 17:41:53 UTC+0000
0xf6b7d8d0	gvfs-fuse-daemo	1257	1000	1000	0x350d2000	2014-05-23 17:41:53 UTC+0000
0xf5be3f70	gvfs-fuse-daemo	1258	1000	1000	0x350d2000	2014-05-23 17:41:53 UTC+0000
0xf50b4c20	compiz	1259	1000	1000	0x351a3000	2014-05-23 17:41:53 UTC+0000
0xf50b32c0	evolution-alarm	1261	1000	1000	0x350e3000	2014-05-23 17:41:54 UTC+0000
0xf506d8d0	polkitd	1266	0	0	0x3511e000	2014-05-23 17:41:54 UTC+0000
0xf5108000	pulseaudio	1267	1000	1000	0x350f2000	2014-05-23 17:41:54 UTC+0000
0xf5108cb0	gvfs-fuse-daemo	1269	1000	1000	0x350d2000	2014-05-23 17:41:54 UTC+0000
0xf510a610	nautilus	1271	1000	1000	0x3511d000	2014-05-23 17:41:54 UTC+0000
0xf510bf70	rtkit-daemon	1273	110	117	0x3512a000	2014-05-23 17:41:54 UTC+0000

Offset	Name	Pid	Uid	Gid	DTB	Start Time
0xf5109960	gnome-panel	1275	1000	1000	0x35126000	2014-05-23 17:41:54 UTC+0000
0xf510e580	nm-applet	1278	1000	1000	0x35131000	2014-05-23 17:41:54 UTC+0000
0xf510cc20	rtkit-daemon	1285	110	117	0x3512a000	2014-05-23 17:41:55 UTC+0000
0xf510b2c0	rtkit-daemon	1286	110	117	0x3512a000	2014-05-23 17:41:55 UTC+0000
0xf5168000	polkit-gnome-au	1287	1000	1000	0x35149000	2014-05-23 17:41:55 UTC+0000
0xf5168cb0	bluetooth-apple	1299	1000	1000	0x35081000	2014-05-23 17:41:56 UTC+0000
0xf516a610	alsa-sink	1302	1000	1000	0x350f2000	2014-05-23 17:41:56 UTC+0000
0xf5169960	alsa-source	1305	1000	1000	0x350f2000	2014-05-23 17:41:56 UTC+0000
0xf50b0cb0	gconf-helper	1310	1000	1000	0x3517d000	2014-05-23 17:41:56 UTC+0000
0xf51c8cb0	evolution-alarm	1317	1000	1000	0x350e3000	2014-05-23 17:41:57 UTC+0000
0xf51c9960	nautilus	1318	1000	1000	0x3511d000	2014-05-23 17:41:57 UTC+0000
0xf51cbf70	gnome-panel	1325	1000	1000	0x35126000	2014-05-23 17:41:57 UTC+0000
0xf516f230	gconf-helper	1334	1000	1000	0x3517d000	2014-05-23 17:41:58 UTC+0000
0xf516bf70	nm-applet	1335	1000	1000	0x35131000	2014-05-23 17:41:58 UTC+0000
0xf516b2c0	gvfsd-trash	1350	1000	1000	0x351e0000	2014-05-23 17:41:59 UTC+0000
0xf5b17230	gvfs-gdu-volume	1368	1000	1000	0x350bf000	2014-05-23 17:42:00 UTC+0000
0xf516cc20	bonobo-activati	1376	1000	1000	0x35210000	2014-05-23 17:42:00 UTC+0000
0xf690cc20	udisks-daemon	1391	0	0	0x35202000	2014-05-23 17:42:01 UTC+0000
0xf516d8d0	gnome-screensav	1395	1000	1000	0x351e9000	2014-05-23 17:42:01 UTC+0000
0xf5230cb0	bonobo-activati	1396	1000	1000	0x35210000	2014-05-23 17:42:01 UTC+0000
0xf5237230	udisks-daemon	1406	0	0	0x351f4000	2014-05-23 17:42:01 UTC+0000
0xf51cb2c0	bonobo-activati	1411	1000	1000	0x35210000	2014-05-23 17:42:02 UTC+0000
0xf5233f70	wnck-applet	1421	1000	1000	0x35249000	2014-05-23 17:42:02 UTC+0000
0xf52332c0	trashapplet	1424	1000	1000	0x35245000	2014-05-23 17:42:02 UTC+0000
0xf5274c20	gvfs-gphoto2-vo	1431	1000	1000	0x351af000	2014-05-23 17:42:02 UTC+0000
0xf5273f70	gvfs-afc-volume	1445	1000	1000	0x350e0000	2014-05-23 17:42:03 UTC+0000
0xf5272610	gvfs-afc-volume	1449	1000	1000	0x350e0000	2014-05-23 17:42:03 UTC+0000
0xf5271960	gdu-notificatio	1452	1000	1000	0x35258000	2014-05-23 17:42:03 UTC+0000
0xf5270cb0	gvfsd-burn	1453	1000	1000	0x351dd000	2014-05-23 17:42:04 UTC+0000
0xf52358d0	indicator-apple	1463	1000	1000	0x352a6000	2014-05-23 17:42:04 UTC+0000
0xf5234c20	clock-applet	1465	1000	1000	0x3523a000	2014-05-23 17:42:04 UTC+0000
0xf50b58d0	notification-ar	1466	1000	1000	0x35268000	2014-05-23 17:42:04 UTC+0000
0xf51ca610	wnck-applet	1467	1000	1000	0x35249000	2014-05-23 17:42:04 UTC+0000
0xf51c8000	trashapplet	1468	1000	1000	0x35245000	2014-05-23 17:42:04 UTC+0000
0xf516e580	indicator-apple	1469	1000	1000	0x351f5000	2014-05-23 17:42:04 UTC+0000
0xf51ccc20	compiz	1474	1000	1000	0x351a3000	2014-05-23 17:42:07 UTC+0000
0xf5232610	indicator-apple	1479	1000	1000	0x352a6000	2014-05-23 17:42:07 UTC+0000
0xf5230000	indicator-apple	1480	1000	1000	0x351f5000	2014-05-23 17:42:07 UTC+0000
0xf50b6580	notification-ar	1481	1000	1000	0x35268000	2014-05-23 17:42:07 UTC+0000
0xf50b7230	clock-applet	1482	1000	1000	0x3523a000	2014-05-23 17:42:07 UTC+0000

Offset	Name	Pid	Uid	Gid	DTB	Start Time		
0xf510f230	gvfsd-metadata	1485	1000	1000	0x3529d000	2014-05-23	17:42:07	UTC+0000
0xf5068000	indicator-sound	1487	1000	1000	0x352a1000	2014-05-23	17:42:07	UTC+0000
0xf5236580	indicator-messa	1489	1000	1000	0x352e2000	2014-05-23	17:42:08	UTC+0000
0xf51cd8d0	sh	1490	1000	1000	0x352f9000	2014-05-23	17:42:08	UTC+0000
0xf5276580	indicator-appli	1492	1000	1000	0x352fa000	2014-05-23	17:42:08	UTC+0000
0xf52758d0	gtk-window-deco	1493	1000	1000	0x35309000	2014-05-23	17:42:08	UTC+0000
0xf510d8d0	gtk-window-deco	1497	1000	1000	0x35309000	2014-05-23	17:42:09	UTC+0000
0xf51cf230	indicator-sessi	1500	1000	1000	0x35329000	2014-05-23	17:42:10	UTC+0000
0xf5348000	indicator-me-se	1502	1000	1000	0x35336000	2014-05-23	17:42:10	UTC+0000
0xf5348cb0	indicator-sessi	1503	1000	1000	0x35329000	2014-05-23	17:42:10	UTC+0000
0xf5349960	indicator-me-se	1504	1000	1000	0x35336000	2014-05-23	17:42:11	UTC+0000
0xf534b2c0	gnome-terminal	1514	1000	1000	0x3535e000	2014-05-23	17:42:23	UTC+0000
0xf534cc20	gnome-terminal	1516	1000	1000	0x3535e000	2014-05-23	17:42:23	UTC+0000
0xf534d8d0	applet.py	1517	1000	1000	0x3536b000	2014-05-23	17:42:23	UTC+0000
0xf534bf70	gnome-pty-helpe	1519	1000	1000	0x35365000	2014-05-23	17:42:24	UTC+0000
0xf534e580	bash	1521	1000	1000	0x35384000	2014-05-23	17:42:24	UTC+0000
0xf5270000	gnome-terminal	1522	1000	1000	0x3535e000	2014-05-23	17:42:25	UTC+0000
0xf5858000	su	1542	0	0	0x3580b000	2014-05-23	17:42:29	UTC+0000
0xf506b2c0	bash	1550	0	0	0x35bea000	2014-05-23	17:42:32	UTC+0000
0xf6928000	update-notifier	1611	1000	1000	0x35076000	2014-05-23	17:42:54	UTC+0000
0xf6b78000	update-notifier	1612	1000	1000	0x35076000	2014-05-23	17:42:55	UTC+0000
0xf6b7e580	flush-vboxsf-1	1614	0	0	----------	2014-05-23	17:42:56	UTC+0000
0xf690b2c0	**_h4x_bd**	**1980**	**2**	**2**	**0x353f6000**	**2014-05-23**	**17:44:58**	**UTC+0000**

B.6 Output for plugin linux_psxview

The following output was generated by the Volatility *linux_psxview* plugin, as found in Section 3.3.6:

Table B.5: Plugin output for linux_psxview (sorted by PID).

Offset(V)	Name	PID	pslist	pid_hash	kmem_cache
0xf7070000	init	1	TRUE	TRUE	FALSE
0xf7070cb0	kthreadd	2	TRUE	TRUE	FALSE
0xf7071960	ksoftirqd/0	3	TRUE	TRUE	FALSE
0xf7072610	migration/0	4	TRUE	TRUE	FALSE
0xf70732c0	watchdog/0	5	TRUE	TRUE	FALSE
0xf7073f70	events/0	6	TRUE	TRUE	FALSE
0xf7074c20	cpuset	7	TRUE	TRUE	FALSE
0xf70758d0	khelper	8	TRUE	TRUE	FALSE
0xf7076580	netns	9	TRUE	TRUE	FALSE
0xf7077230	async/mgr	10	TRUE	TRUE	FALSE
0xf7098000	pm	11	TRUE	TRUE	FALSE
0xf7098cb0	sync_supers	12	TRUE	TRUE	FALSE
0xf7099960	bdi-default	13	TRUE	TRUE	FALSE
0xf709a610	kintegrityd/0	14	TRUE	TRUE	FALSE
0xf709b2c0	kblockd/0	15	TRUE	TRUE	FALSE
0xf709bf70	kacpid	16	TRUE	TRUE	FALSE
0xf709cc20	kacpi_notify	17	TRUE	TRUE	FALSE
0xf709d8d0	kacpi_hotplug	18	TRUE	TRUE	FALSE
0xf709e580	ata_aux	19	TRUE	TRUE	FALSE
0xf709f230	ata_sff/0	20	TRUE	TRUE	FALSE
0xf70f8000	khubd	21	TRUE	TRUE	FALSE
0xf70f8cb0	kseriod	22	TRUE	TRUE	FALSE
0xf70f9960	kmmcd	23	TRUE	TRUE	FALSE
0xf70fb2c0	khungtaskd	25	TRUE	TRUE	FALSE
0xf70fbf70	kswapd0	26	TRUE	TRUE	FALSE
0xf70fcc20	ksmd	27	TRUE	TRUE	FALSE
0xf70fd8d0	aio/0	28	TRUE	TRUE	FALSE
0xf70fe580	ecryptfs-kthrea	29	TRUE	TRUE	FALSE
0xf70ff230	crypto/0	30	TRUE	TRUE	FALSE
0xf73f8000	scsi_eh_0	36	TRUE	TRUE	FALSE
0xf73fa610	scsi_eh_1	37	TRUE	TRUE	FALSE
0xf73fbf70	kstriped	40	TRUE	TRUE	FALSE
0xf73fcc20	kmpathd/0	41	TRUE	TRUE	FALSE
0xf73fd8d0	kmpath_handlerd	42	TRUE	TRUE	FALSE

Offset(V)	Name	PID	pslist	pid_hash	kmem_cache
0xf73fe580	ksnapd	43	TRUE	TRUE	FALSE
0xf73ff230	kondemand/0	44	TRUE	TRUE	FALSE
0xf6878000	kconservative/0	45	TRUE	TRUE	FALSE
0xf696b2c0	scsi_eh_2	155	TRUE	TRUE	FALSE
0xf696a610	usbhid_resumer	164	TRUE	TRUE	FALSE
0xf6968cb0	jbd2/sda1-8	180	TRUE	TRUE	FALSE
0xf6969960	ext4-dio-unwrit	181	TRUE	TRUE	FALSE
0xf6908000	flush-8:0	222	TRUE	TRUE	FALSE
0xf692b2c0	upstart-udev-br	248	TRUE	TRUE	FALSE
0xf692a610	udevd	251	TRUE	TRUE	FALSE
0xf59032c0	udevd	378	TRUE	TRUE	FALSE
0xf5906580	iprt/0	379	TRUE	TRUE	FALSE
0xf5903f70	udevd	383	TRUE	TRUE	FALSE
0xf58858d0	kpsmoused	403	TRUE	TRUE	FALSE
0xf5858cb0	rsyslogd	570	TRUE	TRUE	FALSE
0xf690e580	rsyslogd	575	FALSE	TRUE	FALSE
0xf5883f70	rsyslogd	576	FALSE	TRUE	FALSE
0xf5880cb0	dbus-daemon	581	TRUE	TRUE	FALSE
0xf58e1960	NetworkManager	599	TRUE	TRUE	FALSE
0xf58e2610	avahi-daemon	602	TRUE	TRUE	FALSE
0xf58e0000	avahi-daemon	605	TRUE	TRUE	FALSE
0xf58e7230	modem-manager	610	TRUE	TRUE	FALSE
0xf5907230	wpa_supplicant	630	TRUE	TRUE	FALSE
0xf687f230	dhclient	633	TRUE	TRUE	FALSE
0xf6b78cb0	gdm-binary	636	TRUE	TRUE	FALSE
0xf585e580	NetworkManager	642	FALSE	TRUE	FALSE
0xf58e0cb0	cupsd	653	TRUE	TRUE	FALSE
0xf6bd8000	console-kit-dae	657	TRUE	TRUE	FALSE
0xf59bb2c0	console-kit-dae	658	FALSE	TRUE	FALSE
0xf59b8000	console-kit-dae	659	FALSE	TRUE	FALSE
0xf59bf230	console-kit-dae	660	FALSE	TRUE	FALSE
0xf59bd8d0	console-kit-dae	661	FALSE	TRUE	FALSE
0xf59b8cb0	console-kit-dae	662	FALSE	TRUE	FALSE
0xf59b9960	console-kit-dae	663	FALSE	TRUE	FALSE
0xf59be580	console-kit-dae	664	FALSE	TRUE	FALSE
0xf6968000	console-kit-dae	665	FALSE	TRUE	FALSE
0xf5900cb0	console-kit-dae	666	FALSE	TRUE	FALSE
0xf59058d0	console-kit-dae	667	FALSE	TRUE	FALSE
0xf5904c20	console-kit-dae	668	FALSE	TRUE	FALSE

Offset(V)	Name	PID	pslist	pid_hash	kmem_cache
0xf5902610	console-kit-dae	669	FALSE	TRUE	FALSE
0xf5901960	console-kit-dae	670	FALSE	TRUE	FALSE
0xf581b2c0	console-kit-dae	671	FALSE	TRUE	FALSE
0xf581a610	console-kit-dae	672	FALSE	TRUE	FALSE
0xf6b7f230	console-kit-dae	673	FALSE	TRUE	FALSE
0xf6b7a610	console-kit-dae	674	FALSE	TRUE	FALSE
0xf6b7cc20	console-kit-dae	675	FALSE	TRUE	FALSE
0xf585a610	console-kit-dae	676	FALSE	TRUE	FALSE
0xf585b2c0	console-kit-dae	677	FALSE	TRUE	FALSE
0xf5859960	console-kit-dae	678	FALSE	TRUE	FALSE
0xf6929960	console-kit-dae	679	FALSE	TRUE	FALSE
0xf692e580	console-kit-dae	680	FALSE	TRUE	FALSE
0xf692cc20	console-kit-dae	681	FALSE	TRUE	FALSE
0xf58832c0	console-kit-dae	682	FALSE	TRUE	FALSE
0xf5882610	console-kit-dae	683	FALSE	TRUE	FALSE
0xf5880000	console-kit-dae	684	FALSE	TRUE	FALSE
0xf58e3f70	console-kit-dae	685	FALSE	TRUE	FALSE
0xf5ab8000	console-kit-dae	686	FALSE	TRUE	FALSE
0xf5ab8cb0	console-kit-dae	687	FALSE	TRUE	FALSE
0xf5ab9960	console-kit-dae	688	FALSE	TRUE	FALSE
0xf5aba610	console-kit-dae	689	FALSE	TRUE	FALSE
0xf5abb2c0	console-kit-dae	690	FALSE	TRUE	FALSE
0xf5abbf70	console-kit-dae	691	FALSE	TRUE	FALSE
0xf5abcc20	console-kit-dae	692	FALSE	TRUE	FALSE
0xf5abd8d0	console-kit-dae	693	FALSE	TRUE	FALSE
0xf5abe580	console-kit-dae	694	FALSE	TRUE	FALSE
0xf5abf230	console-kit-dae	695	FALSE	TRUE	FALSE
0xf5ad8000	console-kit-dae	696	FALSE	TRUE	FALSE
0xf5ad8cb0	console-kit-dae	697	FALSE	TRUE	FALSE
0xf5ad9960	console-kit-dae	698	FALSE	TRUE	FALSE
0xf5ada610	console-kit-dae	699	FALSE	TRUE	FALSE
0xf5adb2c0	console-kit-dae	700	FALSE	TRUE	FALSE
0xf5adbf70	console-kit-dae	701	FALSE	TRUE	FALSE
0xf5adcc20	console-kit-dae	702	FALSE	TRUE	FALSE
0xf5add8d0	console-kit-dae	703	FALSE	TRUE	FALSE
0xf5ade580	console-kit-dae	704	FALSE	TRUE	FALSE
0xf5adf230	console-kit-dae	705	FALSE	TRUE	FALSE
0xf5af0000	console-kit-dae	706	FALSE	TRUE	FALSE
0xf5af0cb0	console-kit-dae	707	FALSE	TRUE	FALSE

Offset(V)	Name	PID	pslist	pid_hash	kmem_cache
0xf5af1960	console-kit-dae	708	FALSE	TRUE	FALSE
0xf5af2610	console-kit-dae	709	FALSE	TRUE	FALSE
0xf5af32c0	console-kit-dae	710	FALSE	TRUE	FALSE
0xf5af3f70	console-kit-dae	711	FALSE	TRUE	FALSE
0xf5af4c20	console-kit-dae	712	FALSE	TRUE	FALSE
0xf5af58d0	console-kit-dae	713	FALSE	TRUE	FALSE
0xf5af6580	console-kit-dae	714	FALSE	TRUE	FALSE
0xf5af7230	console-kit-dae	715	FALSE	TRUE	FALSE
0xf5b10000	console-kit-dae	716	FALSE	TRUE	FALSE
0xf5b10cb0	console-kit-dae	717	FALSE	TRUE	FALSE
0xf5b11960	console-kit-dae	718	FALSE	TRUE	FALSE
0xf5b12610	console-kit-dae	719	FALSE	TRUE	FALSE
0xf5b14c20	console-kit-dae	722	FALSE	TRUE	FALSE
0xf5b132c0	gdm-simple-slav	725	TRUE	TRUE	FALSE
0xf5b158d0	gdm-binary	726	FALSE	TRUE	FALSE
0xf58e32c0	Xorg	745	TRUE	TRUE	FALSE
0xf581f230	gdm-simple-slav	747	FALSE	TRUE	FALSE
0xf5be1960	getty	801	TRUE	TRUE	FALSE
0xf5be7230	getty	805	TRUE	TRUE	FALSE
0xf6bdcc20	getty	813	TRUE	TRUE	FALSE
0xf6bda610	getty	815	TRUE	TRUE	FALSE
0xf690a610	getty	817	TRUE	TRUE	FALSE
0xf6908cb0	anacron	822	TRUE	TRUE	FALSE
0xf690bf70	acpid	823	TRUE	TRUE	FALSE
0xf6928cb0	cron	826	TRUE	TRUE	FALSE
0xf692d8d0	atd	827	TRUE	TRUE	FALSE
0xf6b7bf70	libvirtd	839	TRUE	TRUE	FALSE
0xf5819960	libvirtd	845	FALSE	TRUE	FALSE
0xf581cc20	libvirtd	846	FALSE	TRUE	FALSE
0xf581d8d0	libvirtd	847	FALSE	TRUE	FALSE
0xf581bf70	libvirtd	848	FALSE	TRUE	FALSE
0xf5818cb0	libvirtd	849	FALSE	TRUE	FALSE
0xf5884c20	libvirtd	850	FALSE	TRUE	FALSE
0xf5900000	VBoxService	939	TRUE	TRUE	FALSE
0xf5887230	control	941	FALSE	TRUE	FALSE
0xf5b13f70	timesync	942	FALSE	TRUE	FALSE
0xf5b16580	vminfo	943	FALSE	TRUE	FALSE
0xf5be6580	cpuhotplug	944	FALSE	TRUE	FALSE
0xf5be2610	memballoon	945	FALSE	TRUE	FALSE

Offset(V)	Name	PID	pslist	pid_hash	kmem_cache
0xf58e4c20	vmstats	946	FALSE	TRUE	FALSE
0xf6bdb2c0	automount	947	FALSE	TRUE	FALSE
0xf585bf70	gdm-session-wor	965	TRUE	TRUE	FALSE
0xf6bde580	dnsmasq	982	TRUE	TRUE	FALSE
0xf690f230	gnome-session	999	TRUE	TRUE	FALSE
0xf690d8d0	gdm-session-wor	1000	FALSE	TRUE	FALSE
0xf6bdd8d0	VBoxClient	1099	TRUE	TRUE	FALSE
0xf696bf70	SHCLIP	1103	FALSE	TRUE	FALSE
0xf6909960	VBoxClient	1115	TRUE	TRUE	FALSE
0xf585d8d0	X11 monitor	1121	FALSE	TRUE	FALSE
0xf6bdbf70	VBoxClient	1128	TRUE	TRUE	FALSE
0xf692f230	Host events	1131	FALSE	TRUE	FALSE
0xf5818000	VBoxClient	1140	TRUE	TRUE	FALSE
0xf6bd9960	HGCM-NOTIFY	1143	FALSE	TRUE	FALSE
0xf6bd8cb0	X11-NOTIFY	1144	FALSE	TRUE	FALSE
0xf585cc20	ssh-agent	1146	TRUE	TRUE	FALSE
0xf692bf70	dbus-launch	1151	TRUE	TRUE	FALSE
0xf5068cb0	dbus-daemon	1153	TRUE	TRUE	FALSE
0xf506bf70	gnome-session	1178	FALSE	TRUE	FALSE
0xf506e580	gconfd-2	1182	TRUE	TRUE	FALSE
0xf5be0cb0	gnome-power-man	1201	TRUE	TRUE	FALSE
0xf506a610	gnome-session	1203	FALSE	TRUE	FALSE
0xf6b79960	getty	1205	TRUE	TRUE	FALSE
0xf5be32c0	gnome-keyring-d	1210	TRUE	TRUE	FALSE
0xf5be58d0	gnome-keyring-d	1211	FALSE	TRUE	FALSE
0xf5be0000	gnome-keyring-d	1212	FALSE	TRUE	FALSE
0xf506cc20	gnome-power-man	1216	FALSE	TRUE	FALSE
0xf506f230	upowerd	1221	TRUE	TRUE	FALSE
0xf50b2610	gnome-settings-	1226	TRUE	TRUE	FALSE
0xf50b1960	gnome-settings-	1228	FALSE	TRUE	FALSE
0xf50b3f70	gvfsd	1235	TRUE	TRUE	FALSE
0xf58e6580	bluetooth-apple	1255	TRUE	TRUE	FALSE
0xf6bdf230	gvfs-fuse-daemo	1256	TRUE	TRUE	FALSE
0xf6b7d8d0	gvfs-fuse-daemo	1257	FALSE	TRUE	FALSE
0xf5be3f70	gvfs-fuse-daemo	1258	FALSE	TRUE	FALSE
0xf50b4c20	compiz	1259	TRUE	TRUE	FALSE
0xf50b32c0	evolution-alarm	1261	TRUE	TRUE	FALSE

Offset(V)	Name	PID	pslist	pid_hash	kmem_cache
0xf506d8d0	polkitd	1266	TRUE	TRUE	FALSE
0xf5108000	pulseaudio	1267	TRUE	TRUE	FALSE
0xf5108cb0	gvfs-fuse-daemo	1269	FALSE	TRUE	FALSE
0xf510a610	nautilus	1271	TRUE	TRUE	FALSE
0xf510bf70	rtkit-daemon	1273	TRUE	TRUE	FALSE
0xf5109960	gnome-panel	1275	TRUE	TRUE	FALSE
0xf510e580	nm-applet	1278	TRUE	TRUE	FALSE
0xf510cc20	rtkit-daemon	1285	FALSE	TRUE	FALSE
0xf510b2c0	rtkit-daemon	1286	FALSE	TRUE	FALSE
0xf5168000	polkit-gnome-au	1287	TRUE	TRUE	FALSE
0xf5168cb0	bluetooth-apple	1299	FALSE	TRUE	FALSE
0xf516a610	alsa-sink	1302	FALSE	TRUE	FALSE
0xf5169960	alsa-source	1305	FALSE	TRUE	FALSE
0xf50b0cb0	gconf-helper	1310	TRUE	TRUE	FALSE
0xf51c8cb0	evolution-alarm	1317	FALSE	TRUE	FALSE
0xf51c9960	nautilus	1318	FALSE	TRUE	FALSE
0xf51cbf70	gnome-panel	1325	FALSE	TRUE	FALSE
0xf516f230	gconf-helper	1334	FALSE	TRUE	FALSE
0xf516bf70	nm-applet	1335	FALSE	TRUE	FALSE
0xf516b2c0	gvfsd-trash	1350	TRUE	TRUE	FALSE
0xf5b17230	gvfs-gdu-volume	1368	TRUE	TRUE	FALSE
0xf516cc20	bonobo-activati	1376	TRUE	TRUE	FALSE
0xf690cc20	udisks-daemon	1391	TRUE	TRUE	FALSE
0xf516d8d0	gnome-screensav	1395	TRUE	TRUE	FALSE
0xf5230cb0	bonobo-activati	1396	FALSE	TRUE	FALSE
0xf5237230	udisks-daemon	1406	TRUE	TRUE	FALSE
0xf51cb2c0	bonobo-activati	1411	FALSE	TRUE	FALSE
0xf5233f70	wnck-applet	1421	TRUE	TRUE	FALSE
0xf52332c0	trashapplet	1424	TRUE	TRUE	FALSE
0xf5274c20	gvfs-gphoto2-vo	1431	TRUE	TRUE	FALSE
0xf5273f70	gvfs-afc-volume	1445	TRUE	TRUE	FALSE
0xf5272610	gvfs-afc-volume	1449	FALSE	TRUE	FALSE
0xf5271960	gdu-notificatio	1452	TRUE	TRUE	FALSE
0xf5270cb0	gvfsd-burn	1453	TRUE	TRUE	FALSE
0xf52358d0	indicator-apple	1463	TRUE	TRUE	FALSE
0xf5234c20	clock-applet	1465	TRUE	TRUE	FALSE
0xf50b58d0	notification-ar	1466	TRUE	TRUE	FALSE
0xf51ca610	wnck-applet	1467	FALSE	TRUE	FALSE
0xf51c8000	trashapplet	1468	FALSE	TRUE	FALSE

Offset(V)	Name	PID	pslist	pid_hash	kmem_cache
0xf516e580	indicator-apple	1469	TRUE	TRUE	FALSE
0xf51ccc20	compiz	1474	FALSE	TRUE	FALSE
0xf5232610	indicator-apple	1479	FALSE	TRUE	FALSE
0xf5230000	indicator-apple	1480	FALSE	TRUE	FALSE
0xf50b6580	notification-ar	1481	FALSE	TRUE	FALSE
0xf50b7230	clock-applet	1482	FALSE	TRUE	FALSE
0xf510f230	gvfsd-metadata	1485	TRUE	TRUE	FALSE
0xf5068000	indicator-sound	1487	TRUE	TRUE	FALSE
0xf5236580	indicator-messa	1489	TRUE	TRUE	FALSE
0xf51cd8d0	sh	1490	TRUE	TRUE	FALSE
0xf5276580	indicator-appli	1492	TRUE	TRUE	FALSE
0xf52758d0	gtk-window-deco	1493	TRUE	TRUE	FALSE
0xf510d8d0	gtk-window-deco	1497	FALSE	TRUE	FALSE
0xf51cf230	indicator-sessi	1500	TRUE	TRUE	FALSE
0xf5348000	indicator-me-se	1502	TRUE	TRUE	FALSE
0xf5348cb0	indicator-sessi	1503	FALSE	TRUE	FALSE
0xf5349960	indicator-me-se	1504	FALSE	TRUE	FALSE
0xf534b2c0	gnome-terminal	1514	TRUE	TRUE	FALSE
0xf534cc20	gnome-terminal	1516	FALSE	TRUE	FALSE
0xf534d8d0	applet.py	1517	TRUE	TRUE	FALSE
0xf534bf70	gnome-pty-helpe	1519	TRUE	TRUE	FALSE
0xf534e580	bash	1521	TRUE	TRUE	FALSE
0xf5270000	gnome-terminal	1522	FALSE	TRUE	FALSE
0xf5858000	su	1542	TRUE	TRUE	FALSE
0xf506b2c0	bash	1550	TRUE	TRUE	FALSE
0xf6928000	update-notifier	1611	TRUE	TRUE	FALSE
0xf6b78000	update-notifier	1612	FALSE	TRUE	FALSE
0xf6b7e580	flush-vboxsf-1	1614	TRUE	TRUE	FALSE
0xf690b2c0	**_h4x_bd**	**1980**	**TRUE**	**TRUE**	**FALSE**

B.7 Output for plugin linux_lsmod

The following output was generated by the Volatility *linux_lsmod* plugin, as found in Section 3.6.1:

```
nls_cp437 4931
ip6table_filter 1275
        forward=Y
ip6_tables 11764
binfmt_misc 6599
ipt_MASQUERADE 1419
iptable_nat 3752
nf_nat 16289
nf_conntrack_ipv4 10783
        hashsize=16384
nf_defrag_ipv4 1117
xt_state 1014
nf_conntrack 63258
        hashsize=16384
        expect_hashsize=1024
        acct=Y
ipt_REJECT 2004
xt_tcpudp 1927
vboxsf 38249
        follow_symlinks=0
iptable_filter 1302
        forward=Y
ip_tables 10460
x_tables 15921
bridge 68008
stp 1667
snd_intel8x0 25632
        joystick=0
        enable=Y
        spdif_aclink=0
        xbox=Y
        buggy_irq=Y
        buggy_semaphore=Y
        ac97_quirk=(null)
        ac97_clock=0
        id=(null)
        index=-1
```

```
snd_ac97_codec 99227
        power_save=0
        enable_loopback=Y
ac97_bus 1014
snd_pcm 71475
        maximum_substreams=4
        preallocate_dma=1
snd_seq_midi 4588
        input_buffer_size=4096
        output_buffer_size=4096
snd_rawmidi 17783
snd_seq_midi_event 6047
snd_seq 47174
        seq_default_timer_resolution=0
        seq_default_timer_subdevice=0
        seq_default_timer_device=3
        seq_default_timer_card=-1
        seq_default_timer_sclass=0
        seq_default_timer_class=1
        seq_client_load=14,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1
snd_timer 19067
        timer_tstamp_monotonic=1
        timer_limit=1
snd_seq_device 5744
ppdev 5556
vboxvideo 1279
parport_pc 26058
        init_mode=(null)
        verbose_probing=0
        dma=(null),(null),(null),(null),(null),(null),(null),(null),(null),(null),(null),(null),(
null),(null),(null),(null)
        irq=(null),(null),(null),(null),(null),(null),(null),(null),(null),(null),(null),(null),(nu
ll),(null),(null),(null)
        io_hi=,,,,,,,,,,,,,,,,
        io=,,,,,,,,,,,,,,,,
drm 168054
        debug=0
snd 49006
        cards_limit=1
        major=116
```

slots=(null),(null)
Unknown get_fn: 0xf8033130
psmouse 59033
resync_time=0
resetafter=5
smartscroll=Y
rate=100
resolution=200
proto=None
force_elantech=Y
post_interrupt_delay=1000
recal_guard_time=2000
spew_delay=1000
jumpy_delay=1000
recalib_delta=100
tpdebug=Y
serio_raw 4022
joydev 8735
agpgart 32011
Unknown get_fn: 0xf81c9260
Unknown get_fn: 0xf81c9190
Unknown get_fn: 0xf81c90c0
vboxguest 211970
log_dest=None
log_flags=None
log=None
i2c_piix4 8635
force_addr=0
force=0
soundcore 880
snd_page_alloc 7120
lp 7342
reset=Y
parport=(null),(null),(null),(null),(null),(null),(null),(null)
parport 31492
usbhid 36882
quirks=(null),(null),(null),(null)
ignoreled=0
mousepoll=0
hid 67742

```
        debug=0
e1000 97525
        debug=3
        copybreak=256
        SmartPowerDownEnable=
        InterruptThrottleRate=
        RxAbsIntDelay=
        RxIntDelay=
        TxAbsIntDelay=
        TxIntDelay=
        XsumRX=
        FlowControl=
        AutoNeg=
        Duplex=
        Speed=
        RxDescriptors=
        TxDescriptors=
ahci 19013
        marvell_enable=0
libahci 21667
        ahci_em_messages=1
        ignore_sss=0
        skip_host_reset=0
```

Bibliography

Carbone, R. File recovery and data extraction using automated data recovery tools: A balanced approach using Windows and Linux when working with an unknown disk image and filesystem. TM 2009-161. Technical memorandum. DRDC – Valcartier Research Centre. January 2013.

Carbone, R. Malware memory analysis for non-specialists: Investigating a publicly available memory image of the Zeus Trojan horse. TM 2013-018. Technical memorandum. DRDC – Valcartier Research Centre. April 2013.

Carbone, R. Malware memory analysis for non-specialists: Investigating publicly available memory images for Prolaco and SpyEye. TM 2013-155. Technical memorandum. DRDC – Valcartier. October 2013.

Carbone, R. Malware memory analysis for non-specialists: Investigating publicly available memory image 0zapftis (R2D2). TM 2013-177. Technical memorandum. DRDC – Valcartier Research Centre. October 2013.

Carbone, R. Malware memory analysis for non-specialists: Investigating publicly available memory image for the Stuxnet worm. SR DRDC-RDDC-2013-R1. Scientific report. DRDC – Valcartier Research Centre. November 2013.

Carbone, R. Malware memory analysis for non-specialists: Investigating publicly available memory image for the Tigger Trojan horse. SR DRDC-RDDC-2014-R28. Scientific report. DRDC – Valcartier Research Centre. June 2014.

List of symbols/abbreviations/acronyms/initialisms

ACPI	Advanced Configuration and power Interface
APIC	Advanced Programmable Interrupt Controller
AV	Anti-virus or antivirus
BIOS	Basic Input / Output System
CAF	Canadian Armed Forces
CFNOC	Canadian Forces Network Operation Centre
CPU	Central Processing Unit
DRDC	Defence Research and Development Canada
DSTKIM	Director Science and Technology Knowledge and Information Management
DTB	Directory Table Base
DVD	Digital Video Disc or Digital Versatile Disc
DVD +/- RW	Digital Video Disc +/- Read/Write
ECL	Export Control List
ELF	Executable and Linkable Format
FAC	Forces armées canadiennes
FD	File Descriptor
GB	Gigabyte (1x109)
GCC	GNU C Compiler
GHz	Gigahertz
GiB	Gibibyte (230 bytes)
GID	Group ID
Gbps	Gigabits per second
GRC	Gendarmerie Royale du Canada
ID	Identification
IDT	Interrupt Descriptor Table
IT	Information Technology
ITCU	Integrated Technological Crime Unit
KiB	Kibibyte (210 bytes)
LKM	Loadable Kernel Module

MD5	Message-Digest Algorithm 5
NSRL	National Software Reference Library
PAE	Physical Address Extension
PC	Personal Computer
PCI	Peripheral Component Interconnect
PID	Process ID
PO Box	Post-Office Box or Post Office Box
PPID	Parent Process ID
R&D	Research & Development
RAM	Random Access Memory
RCMP	Royal Canadian Mounted Police
ROM	Read-Only Memory
SATA	Serial ATA or Serial AT Attachment or
SHA1	Secure Hash Algorithm-1
SMP	Symmetric Multiprocessing
Syscall	System Call
TCP	Transmission Control Protocol
TI	Technologie d'information
TM	Technical Memorandum
UDP	User Datagram Protocol
UID	User ID
UTC	Coordinated Universal Time
VM	Virtual Machine
VM	Virtual Machine
x64	64-bit PC architecture
x86	32-bit PC architecture

This page intentionally left blank.

DOCUMENT CONTROL DATA

(Security markings for the title, abstract and indexing annotation must be entered when the document is Classified or Designated)

1. ORIGINATOR (The name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g. Centre sponsoring a contractor's report, or tasking agency, are entered in section 8.)

Defence Research and Development Canada – Valcartier
2459 Pie-XI Blvd North
Quebec (Quebec)
G3J 1X5 Canada

2a. SECURITY MARKING (Overall security marking of the document including special supplemental markings if applicable.)

UNCLASSIFIED

2b. CONTROLLED GOODS

(NON-CONTROLLED GOODS)
DMC A
REVIEW: GCEC APRIL 2011

3. TITLE (The complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S, C or U) in parentheses after the title.)

Memory analysis of the KBeast Linux rootkit : Investigating publicly available Linux rootkit using the Volatility memory analysis framework

4. AUTHORS (last name, followed by initials – ranks, titles, etc. not to be used)

Carbone, R.

5. DATE OF PUBLICATION (Month and year of publication of document.)	6a. NO. OF PAGES (Total containing information, including Annexes, Appendices, etc.)	6b. NO. OF REFS (Total cited in document.)
June 2015	90	16

7. DESCRIPTIVE NOTES (The category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.)

Scientific Report

8. SPONSORING ACTIVITY (The name of the department project office or laboratory sponsoring the research and development – include address.)

Defence Research and Development Canada – Valcartier
2459 Pie-XI Blvd North
Quebec (Quebec)
G3J 1X5 Canada

9a. PROJECT OR GRANT NO. (If appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant.)	9b. CONTRACT NO. (If appropriate, the applicable number under which the document was written.)
31XF20 « MOU RCMP "Live Forensics" »	

10a. ORIGINATOR'S DOCUMENT NUMBER (The official document number by which the document is identified by the originating activity. This number must be unique to this document.)	10b. OTHER DOCUMENT NO(s). (Any other numbers which may be assigned this document either by the originator or by the sponsor.)
DRDC-RDDC-2015-R064	

11. DOCUMENT AVAILABILITY (Any limitations on further dissemination of the document, other than those imposed by security classification.)

Unlimited

12. DOCUMENT ANNOUNCEMENT (Any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, where further distribution (beyond the audience specified in (11) is possible, a wider announcement audience may be selected.))

Unlimited

13. ABSTRACT (A brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), (R), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual.)

This report is the first in a series examining Linux Volatility-specific memory malware-based analysis techniques. With minimal use of scanner-based technologies, the author will demonstrate what to look for while conducting Linux-based memory investigations using Volatility. This investigation consists of a memory image infected by the KBeast rootkit that will be analysed using Volatility. Through the proper application of various Volatility plugins combined with an in-depth knowledge of the Linux operating system, this case study can provide guidance to other investigators in their own Linux-based memory analyses.

Ce rapport est le premier d'une série examinant les techniques spécifiques d'analyse de logiciels malveillants en mémoire sous Linux à l'aide de l'outil Volatility. En utilisant au minimum les technologies basées sur des scans, l'auteur démontrera ce qu'il faut rechercher lors d'une investigation de la mémoire Linux avec Volatility. Cette investigation avec Volatility sera effectuée sur une image mémoire infectée par le programme malveillant furtif KBeast. Nous espérons qu'en combinant correctement les différents greffons de Volatility et avec une connaissance approfondie du système d'exploitation Linux, cette étude de cas servira de guide à d'autres investigateurs dans leur propre analyse de mémoire Linux.

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (Technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus, e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

Anti-virus; Antivirus; Computer forensics; Computer infection; Computer memory forensics; Digital forensics; Digital memory forensics; Forensics; Infection; KBeast; Linux; Malware; Memory analysis; Memory forensics; Memory image; Rootkit; Scanners; Virus scanner; Volatility